NETCONF
Internet-Draft
Intended status: Standards Track
Expires: 29 July 2022

B. Claise
Huawei
M. Nayyar
A. Reddy Sesani
Cisco Systems, Inc.
25 January 2022

Per-Node Capabilities for Optimum Operational Data Collection
draft-claise-netconf-metadata-for-collection-03

Abstract

This document proposes a YANG module that provides per-node capabilities for optimum operational data collection. This YANG module augments the YANG Modules for describing System Capabilities and YANG-Push Notification capabilities.

This module defines augmented nodes to publish the metadata information specific to YANG node-identifier as per ietf-system-capabilities datatree.

Complementary RPCs, based on the same node capabilities, simplify the data collection operations.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 29 July 2022.

Copyright Notice

Table of Contents

1. Terminology 2
2. Introduction 3
3. Concepts 4
4. Base ietf-system-node-metadata YANG module 7
4.1. Tree View 7
4.2. Full Tree View 8
4.3. YANG Module 9
5. Examples 15
6. Security Considerations 21
7. IANA Considerations 21
7.1. The IETF XML Registry 21
8. Open Issues 22
9. References 22
9.1. Normative References 22
9.2. Informative References 23
Acknowledgements 24
Authors' Addresses 24

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The term Client and Server are specified in [RFC8342].

The term Implementation-time and Run-time are specified in [I-D.ietf-netconf-notification-capabilities].

2. Introduction

This document specifies a way to learn from the devices how granular its telemetry and data can be to provide the best post-processing analytics. In the end, the service assurance architecture [I-D.claise-opsawg-service-assurance-architecture], it's not sufficient to simply stream (or poll) telemetry data, it is equally important to be able to act on the data. As such, a series of extra information about the node capabilities is essential.

The module ietf-system-capabilities [I-D.ietf-netconf-notification-capabilities] provides a structure that can be used to specify YANG related system capabilities for servers. The module can be used in conjunction with YANG Instance Data to make this information available at implementation-time. The module can also be used to report capability information from the server at run-time.

The module ietf-notification-capabilities [I-D.ietf-netconf-notification-capabilities] augments ietf-system-capabilities to specify capabilities related to "Subscription to YANG Datastores" (YANG-Push) [RFC8639]. It provides a starting point by specifying some per-node telemetry-related capabilities. Of particular interest are the following node capabilities:

* minimum-dampening-period

* on-change-supported

* periodic-notifications-supported

* supported-excluded-change-type

Taking the example of on-change-supported and periodic-notifications-supported, it's key to understand whether a publisher is capable of sending on-change notifications versus sending periodic notifications for the selected data store or data nodes. Indeed, not only would the telemetry configuration change depending on the capabilities (on-change versus periodic), but more importantly the client's handling of the telemetry information would change. Upon receipt of an on-change telemetry message, an immediate action could be taken to correct or mitigate the issue, while in case of periodic notification, a comparison with the previous value must first be performed in order to understand if and how the network state has changed.

Exactly like a client that connects to a server is able to discover the capabilities in terms of supported YANG modules, features, deviations, and protocol capabilities; the same client must also be able to discover the required per-node capabilities (also known as metadata) to correctly act on the telemetry information. It forms part of the API contract for managing and monitoring the device. Extending the per-node capabilities specified in [I-D.ietf-netconf-notification-capabilities], additional per-node capabilities are required.

The YANG module in this document augments the ietf-system-capabilities YANG module in "YANG Modules for describing System Capabilities and Yang-Push Notification Capabilities" [I-D.ietf-netconf-notification-capabilities].

The YANG data model in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

3. Concepts

Doing networking data collection for the sake of doing collection is not useful. At the time of network automation, displaying nice graphs from collected data is not useful: the collected data must be acted upon immediately. Some use cases are: network availability, closed loop automation (reconfiguring network based on observed network state changes), service assurance [I-D.claise-opsawg-service-assurance-architecture], etc.

Along with the capabilities specified in ietf-netconf-notification-capabilities [I-D.ietf-netconf-notification-capabilities] YANG model, there is some additional information that can be made available per node-selector to help with this optimum collection of operational data. For example, these additional metadata can help reduce the load on the devices being managed along with the performance improvements because of the way data is subscribed to. Some other metadata can help with the collection automation itself (mapping of config and oper data node, mapping of MIB oid to YANG leaf).

Some metadata are static and can augment the node-capabilities in [I-D.ietf-netconf-notification-capabilities], for both implementation time and run time environments. Other metadata are dynamic and have to be derived during the run-time. They can change based on the role of the device and the scale of the data being observed.

Per-node static metadata includes:

* minimum-observable-period: This is the minimum observable period in nanoseconds for the node-selector. Streaming or polling more frequently then this interval may not fetch useful information as the node could be updated only at this frequency internally. If a close loop automation system would stream or poll more frequently, it could actually draw the wrong conclusions. Let's take the example of interface counters than are updated more frequently than 30 seconds in a distributed system. Streaming interface counters every 30 seconds would see an natural increase in the interface counters. However, streaming those interface counters every 10 seconds could lead to the wrong conclusion that no packets are received/sent on that specific interface ... triggering an automatic interface troubleshooting action. Hence determining the minimum-observable-period for every monitored leaf is essential for closed loop automation and assurance systems.

* suggested-observable-period: The suggested observable period for this node-selector. This value represents factory default suggested information, only available at implementation time. Let us assume that an assurance system would like to monitor all FIB entries in the router. The router would advertise that the suggested observable period is, let's say, 30 seconds. Those 5 seconds are the factory defaults, provided at the implementation time. Once the router is in production, the observable period would obviously change depending on the environment (as an example, a FIB containing all BGP entries is huge): this dynamic suggest observable period is called the computed-observable-period and is available part of the get-measurement-metadata RPC.

* optimized-measurement-point: In some server design, operational data are usually modeled/structured in a way that the relevant data are grouped together and reside together. In most cases, it is more performant to fetch this data together than as individual leaves: data are structured together internally, grouped together, and therefore fetched together. This feature specifies optimum observable points in the model at which data can be collected and streamed in an efficient way. Depending on the implementation, optimum points can be leaves or a container nodes in the YANG tree. This is a selection node, that means its presence for a node-selector indicates it is the optimized measurement point.

* corresponding-mib-oid: The object identifier (OID) assigned to a SMIv2 definition, corresponding to the node-selector. The object identifier value is written in decimal dotted notation. Existing SNMP MIBs based automations can use this information to migrate to more analytics-ready YANG Modeled data. Working from a single data model system (YANG-based in this case) for data collection simplifies the management, as opposed to use different data

models. Therefore, knowing the mapping MIB OID/YANG leaf is important, as transition mechanism towards YANG (for example: moving away from SNMP polling to model-driven telemetry) but also as a way to understand whether the same operational data is metered in both the MIB and YANG worlds, adding to the load on devices. Some IETF RFCs, such as the YANG Interface Management [RFC8343], specify the mapping in the document. However, providing this mapping directly from the server helps automation from a client point of view.

* related-node: Data nodes that are related for closed-loop scenario for data node specified in node-capabilities. In case node-capabilities is an operational node then the associated node-instance-identifier represents config paths directly related to this operational node capabilities. In case node-capabilities is an config node then the associated node-instance-identifier represents operational leaf directly related to this configuration node capabilities. This node is specifically interesting for non NMDA [RFC8342], non openconfig YANG modules. For example, in the initial YANG data model for interface management [RFC7223], which is not NMDA-compliant, advertising the mapping between the admin-status and the oper-status leaves would clearly simplify the closed loop automation. Note that NMDA and the openconfig -state container solved that issue but not all servers are NMDA compliant and openconfig models don’t cover all server functions.

A generic RPC, get-system-node-capabilities, provides the capabilities for the nodes in the subtree of the input. If the input node passed is a leaf/leaf-list, then all the metadata for that input node are returned. If the input node is not leaf/leaf-list then the RPC returns the metadata of all of its subtree nodes.

There is some run-time information that is very helpful for the applications to know, to be able to start listening to the device without adding too much additional resource strain on the device. The get-measurement-metadata RPC can be used to fetch this data.

Per-node dynamic metadata includes, part of the get-measurement-metadata RPC:

* optimized-measurement-point: The node-selector is searched up the data tree chain to find the parent node that is the optimized measurement point (if the optimized-measurement-point-feature is supported). If the node-selector itself is the optimized point then same data node is returned in the output. If the node-selector has no optimized measurement point then this optimized-measurement-point leaf is not returned.

* computed-observable-period: the computed observable period for this node-selector (and optimized-measurement-point). The system internally dynamically computes the suggested observable period (relevant for polling or streaming cadence) which can be greater-or-equal to the minimal-observable-period. Since this value is dynamic, this metadata is only available in a run time environment.

* active-measurements - subscribed-measurement-period: List of existing subscriptions for this node-selector. If there are no active subscriptions then system calculate the measurement-period and this list is not-returned, else, each instance in this list will be pair of active measurement with intended and actual period used by the system.

4. Base ietf-system-node-metadata YANG module

4.1. Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-system-node-metadata data model.

```
module: ietf-system-node-metadata
  augment /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-cap
abilities/sysc:node-selection/sysc:node-selector:
    +--ro minimum-observable-period?    uint64
    +--ro suggested-observable-period?   uint64
    +--ro optimized-measurement-point?   empty {optimized-measurement-point-featu
re}?
    +--ro corresponding-mib-oid?         yang:object-identifier-128
    +--ro related-node?                  yang:node-instance-identifier

  rpcs:
    +---x get-measurement-metadata
    |  +---w input
    |  |  +---w node-selector?   yang:node-instance-identifier
    |  +--ro output
    |     +--ro optimized-measurement-point?   yang:node-instance-identifier {opt
imized-measurement-point-feature}?
    |     +--ro computed-observable-period?    uint64
    |     +--ro active-measurements* []
    |        +--ro subscribed-measurement-period?   uint64
    +---x get-system-node-capabilities
       +---w input
       |  +---w node-selector?   yang:node-instance-identifier
       +--ro output
          +--ro node-selector-capability* []
             +--ro node?                           yang:node-instance-identifier
             +--ro minimum-observable-period?    uint64
             +--ro suggested-observable-period?   uint64
             +--ro optimized-measurement-point?   empty {optimized-measurement-po
int-feature}?
             +--ro corresponding-mib-oid?         yang:object-identifier-128
             +--ro related-node?                  yang:node-instance-identifier
```

4.2. Full Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-system-capabilities and ietf-system-node-metadata data models.

```
module: ietf-system-node-metadata

  rpcs:
    +---x get-measurement-metadata
    |  +---w input
    |  |  +---w node-selector?   yang:node-instance-identifier
    |  +--ro output
    |     +--ro optimized-measurement-point?   yang:node-instance-identifier {opt
imized-measurement-point-feature}?
    |     +--ro computed-observable-period?    uint64
    |     +--ro active-measurements* []
    |        +--ro subscribed-measurement-period?   uint64
    +---x get-system-node-capabilities
       +---w input
       |  +---w node-selector?   yang:node-instance-identifier
       +--ro output
          +--ro node-selector-capability* []
             +--ro node?                            yang:node-instance-identifier
             +--ro minimum-observable-period?    uint64
             +--ro suggested-observable-period?   uint64
             +--ro optimized-measurement-point?   empty {optimized-measurement-po
int-feature}?
             +--ro corresponding-mib-oid?         yang:object-identifier-128
             +--ro related-node?                  yang:node-instance-identifier
module: ietf-system-capabilities
  +--ro system-capabilities
     +--ro datastore-capabilities* [datastore]
        +--ro datastore                -> /yanglib:yang-library/datastore/name
        +--ro per-node-capabilities* []
           +--ro (node-selection)?
              +--:(node-selector)
                 +--ro node-selector?                                  nacm:node-inst
ance-identifier
                 +--ro sys-metadata:minimum-observable-period?    uint64
                 +--ro sys-metadata:suggested-observable-period?   uint64
                 +--ro sys-metadata:optimized-measurement-point?   empty {optimiz
ed-measurement-point-feature}?
                 +--ro sys-metadata:corresponding-mib-oid?         yang:object-id
entifier-128
                 +--ro sys-metadata:related-node?                  yang:node-inst
ance-identifier
```

4.3. YANG Module

```
   <CODE BEGINS> file "ietf-system-node-metadata@2020-03-20.yang"

   module ietf-system-node-metadata {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-system-node-metadata";
     prefix sys-metadata;

     import ietf-system-capabilities {
       prefix sysc;
       reference
```

```
        "RFC XXXX: YANG Modules for describing System Capabilities and
         Yang-Push Notification Capabilities";
    }
    import ietf-yang-types {
      prefix yang;
      reference
        "RFC XXXX: Currently draft-ietf-netmod-rfc6991-bis-04, Common
         YANG Data types";
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group";
    contact
      "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
       WG List:  <mailto:netconf@ietf.org>

       Editor:   Benoit Claise
                 <mailto:bclaise@cisco.com>

       Editor:   Munish Nayyar
                 <mailto:mnayyar@cisco.com>

       Editor:   Adithya Reddy Sesani
                 <mailto:adithyas@cisco.com>
      ";
    description
      "This document proposes a YANG module that provides per-node
       capabilities for optimum operational data collection.

       This YANG module augments the YANG Modules for describing
       System Capabilities and Yang-Push Notification capabilities
       [RFC XXXX].

       This module defines augmented nodes to publish the
       metadata information specific to YANG node-identifier as per
       ietf-system-capabilities datatree.

       Complementary RPCs, based on the same node capabilities,
       simplify the data collection operations.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
       'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
       'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
       are to be interpreted as described in BCP 14 (RFC 2119)
       (RFC 8174) when, and only when, they appear in all
       capitals, as shown here.

       Copyright (c) 2020 IETF Trust and the persons identified as
```

```
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
        for full legal notices.";

    revision 2020-03-23 {
      description
        "Initial version";
      reference
        "RFC XXX: Per-Node Capabilities For Closed Loop Automation.";
    }

    feature optimized-measurement-point-feature {
      description
        "Support for optimized measurement point within data tree.";
    }

    grouping system-node-metadata-info {
      description
        "group of metadata properties associated to the
         node-instance.";
      leaf minimum-observable-period {
        type uint64;
        units "nanoseconds";
        description
          "The minimum observable period for this node-selector. Don't
           poll or stream more frequently that minimum observable
           period in nanoseconds as the corresponding counter is not
           updated more frequently.";
      }
      leaf suggested-observable-period {
        type uint64;
        units "nanoseconds";
        description
          "The suggested observable period for this node-selector.
           This value represents factory default suggested
           information, only available at implementation time.";
      }
      leaf optimized-measurement-point {
        if-feature "optimized-measurement-point-feature";
```

```
        type empty;
        description
          "This node-selector is an optimized measurement point.";
      }
      leaf corresponding-mib-oid {
        type yang:object-identifier-128;
        description
          "The object identifier (OID) assigned to a SMIv2 definition,
           corresponding to this node-selector.";
      }
      leaf related-node {
        type yang:node-instance-identifier;
        description
          "In case the node instance is an operational node then the
           associated node-instance-identifier represents the config
           leaf directly related to this operational node. In case the
           node instance is an config node then the associated
           node-instance-identifier represents the operational leaf
           directly related to this configuration node. A typical
           example is the relationship between the admin-status and
           oper-status, which is impossible to detect automatically in
           a non-NMDA environment or for non-openconfig YANG moduels.
           The related-node SHOULD NOT reported for NMDA architectures
           and openconfig YANG modules.";
      }
    }

    augment
      "/sysc:system-capabilities/sysc:datastore-capabilities/"
       + "sysc:per-node-capabilities/"
       + "sysc:node-selection/sysc:node-selector" {
      description
        "Metadata information tied to the per-node-capabilities";
      uses system-node-metadata-info;
    }

    rpc get-measurement-metadata {
      description
        "RPC that returns the optimized measurement per-node
         capabilities and some measurement parameters. This RPC
         is added to allow clients to learn dynamically changing
         metadata for a specific leaf on a server.

         If the server supports the optimized-measurement-point
         feature, then the output data refers to
         optimized-measurement-point. The server will internally
         find the optimized-measurement-point. If it can not find it,
         then no output is returned (for the
```

```
          optimized-measurement-point, computed-observable-period,
          and active-measurements).

          If the server doesn't support the optimized-measurement-point
          feature, then the output data refers to input node selector.";
        input {
          leaf node-selector {
            type yang:node-instance-identifier;
            description
              "node instance for which metadata is requested";
          }
        }
        output {
          leaf optimized-measurement-point {
            if-feature "optimized-measurement-point-feature";
            type yang:node-instance-identifier;
            description
              "The node-selector is searched up the data tree chain to
               find the parent node that is the optimized measurement
               point (if the optimized-measurement-point-feature is
               supported).

               If the node-selector itself is the optimized point then
               same data node is returned in the output.

               If the node-selector has no optimized measurement point
               then this optimized-measurement-point leaf is not
               returned.";
          }
          leaf computed-observable-period {
            type uint64;
            units "nanoseconds";
            description
              "the computed observable period for this node-selector (and
               optimized-measurement-point). The system internally
               dynamically computes the suggested observable period
               (relevant for polling or streaming cadence) which can be
               greater-or-equal to the minimal-observable-period.
               Since this value is dynamic, this metadata is only
               available in a run time environment.";
          }
          list active-measurements {
            description
              "list of existing subscriptions for this node-selector. If
               there are no active subscriptions then system calculate
               the measurement-period and this list is not-returned,
               else, each instance in this list will be pair of active
               measurement with intended and actual period used by the
```

```
            system";
          leaf subscribed-measurement-period {
            type uint64;
            units "nanoseconds";
            description
              "Currently subscribed measurement period for this
               node-selector (and optimized-measurement-point)";
          }
        }
      }
    }

    rpc get-system-node-capabilities {
      description
        "RPC to get the capabilities for the nodes in the subtree of
         the input.
         If the input node passed is a leaf/leaf-list, then
         the same node metadata is returned in the output.
         If the input node is not leaf/leaf-list then metadata of its
         subtree nodes is returned.";
      input {
        leaf node-selector {
          type yang:node-instance-identifier;
          description
            "node instance whose subtree which metadata is requested.";
        }
      }
      output {
        list node-selector-capability {
          description
            "metadata of nodes in the subtree of node-selector.";
          leaf node {
            type yang:node-instance-identifier;
            description
              "instance path of the node inside subtree of
               node-selector.";
          }
          uses system-node-metadata-info;
        }
      }
    }
  }

  <CODE ENDS>
```

5. Examples

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

XML data tree for the ietf-interface YANG module [RFC8343]:

```
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name/>
      <description/>
      <type/>
      <link-up-down-trap-enable/>
      <admin-status/>
      <oper-status/>
      <last-change/>
      <if-index/>
      <phys-address/>
      <higher-layer-if>
        <!-- # entries: 0.. -->
      </higher-layer-if>
      <lower-layer-if>
        <!-- # entries: 0.. -->
      </lower-layer-if>
      <speed/>
      <statistics>
        <discontinuity-time/>
        <in-octets/>
        <in-unicast-pkts/>
        <in-broadcast-pkts/>
        <in-multicast-pkts/>
        <in-discards/>
        <in-errors/>
        <in-unknown-protos/>
        <out-octets/>
        <out-unicast-pkts/>
        <out-broadcast-pkts/>
        <out-multicast-pkts/>
        <out-discards/>
        <out-errors/>
      </statistics>
    </interface>
  </interfaces>
```

Example1: Demonstrating the querying metadata for all system schema nodes for the ietf-interfaces [RFC8343].

```
  <!-- Request -->
 <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get>
    <filter>
      <system-capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabil
ities">
      </system-capabilities>
    </filter>
  </get>
</rpc>

  <!-- Response -->
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
  <system-capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilitie
s" xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <datastore-capabilities>
      <datastore>ds:operational</datastore>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface</node-selector>
        <optimized-measurement-point xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata"/>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata"></corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:admin-status</node-selector
>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.7</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:oper-status</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.8</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:if-index</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.1</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
```

em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:phys-address</node-selector
>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.6</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>

```
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:lower-layer-if</node-select
or>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.2.1.2</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:higher-layer-if</node-selec
tor>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.2.1.1</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:speed</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.5</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics</node-selector>
        <optimized-measurement-point xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata"/>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.1</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:discontinuity
-time</node-selector>
        <optimized-measurement-point xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata"/>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.1.1.19</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:in-octets</no
de-selector>
```

```
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.10</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:in-unicast-pk
ts</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.11</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:in-multicast-
pkts</node-selector>
```

```
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.1.1.2</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:in-broadcast-
pkts</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.1.1.3</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:in-discards</
node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.13</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:in-errors</no
de-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.14</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:in-unknown-pr
otos</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.15</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:out-octets</n
ode-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.16</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
```

```
        <node-selector>/if:interfaces/if:interface/if:statistics/if:out-unicast-p
kts</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.17</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:out-multicast
-pkts</node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.1.1.4</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:out-broadcast
-pkts</node-selector>
```

```
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.31.1.1.1.5</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:out-discards<
/node-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.19</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics/if:out-errors</n
ode-selector>
        <corresponding-mib-oid xmlns="urn:ietf:params:xml:ns:yang:ietf-system-nod
e-metadata">1.3.6.1.2.1.2.2.1.20</corresponding-mib-oid>
        <minimum-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-system
-node-metadata">1000</minimum-observable-period>
        <suggested-observable-period xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata">1000</suggested-observable-period>
      </per-node-capabilities>
    </datastore-capabilities>
  </system-capabilities>
</data>
</rpc-reply>
```

Example2: Demonstrating the querying metadata of all optimized-measurement-point(s). Use containment and selection nodes filtering criteria to express which all metadata you want. In this example: get query filter only to "select" the node-instance-identifier, optimized-measurement-point nodes, for the ietf-interfaces [RFC8343]. There are two optimized-measurement-points: interface and statistics.

```
   <!-- Request -->
 <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
     <filter type="subtree">
     <system-capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabili
ties"></system-capabilities>
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
        <optimized-measurement-point xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata"/>
        </per-node-capabilities>
      </datastore-capabilities>
    </filter>
   </get>
 </rpc>

  <!-- Response -->

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
  <system-capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilitie
s" xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <datastore-capabilities>
      <datastore>ds:operational</datastore>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface</node-selector>
        <optimized-measurement-point xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata"/>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>/if:interfaces/if:interface/if:statistics</node-selector>
        <optimized-measurement-point xmlns="urn:ietf:params:xml:ns:yang:ietf-syst
em-node-metadata"/>
      </per-node-capabilities>
    </datastore-capabilities>
  </system-capabilities>
</data>
</rpc-reply>
```

   Example3: Demonstrating the usage of RPC to query the device for
   computed-measurement-period and the subscribed-measurement-period(s)
   for the in-errors YANG leaf.

```
<!-- Request -->

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get-measurement-metadata xmlns="urn:ietf:params:xml:ns:yang:ietf-system-node-m
etadata">
    <node-selector>/if:interfaces/if:interface/if:statistics/if:in-errors</node-s
elector>
  </get-measurement-metadata>
</rpc>

<!-- Response -->
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <optimized-measurement-point>/if:interfaces/if:interface/if:statistics</optimiz
ed-measurement-point>
  <computed-measurement-period>3000</computed-measurement-period>
  <active-measurements>
    <subscribed-measurement-period>1000</subscribed-measurement-period>
  </active-measurements>
  <active-measurements>
    <subscribed-measurement-period>1000</subscribed-measurement-period>
  </active-measurements>
  <active-measurements>
    <subscribed-measurement-period>1000</subscribed-measurement-period>
  </active-measurements>
</rpc-reply>
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

7. IANA Considerations

7.1. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-system-node-metadata
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

8. Open Issues

"related-node" should be split into two: "related-config-node" and "related-state-node"?

Explain how to use the RPC from the client side, along with the different options.

Expand on the active measurement use case

nanosecond: an overkill?

security considerations: see https://trac.ietf.org/trac/ops/wiki/yang-security-guidelines

9. References

9.1. Normative References

[I-D.ietf-netconf-notification-capabilities]
Lengyel, B., Clemm, A., and B. Claise, "YANG Modules describing Capabilities for Systems and Datastore Update Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-notification-capabilities-21, 15 October 2021, <https://www.ietf.org/archive/id/draft-ietf-netconf-notification-capabilities-21.txt>.

[I-D.ietf-netmod-rfc6991-bis]
Schoenwaelder, J., "Common YANG Data Types", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc6991-bis-10, 14 January 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-rfc6991-bis-10.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

9.2. Informative References

[I-D.claise-opsawg-service-assurance-architecture] Claise, B., Quilbeuf, J., Lopez, D. R., Voyer, D., and T. Arumugam, "Service Assurance for Intent-based Networking Architecture", Work in Progress, Internet-Draft, draft-claise-opsawg-service-assurance-architecture-05, 23 April 2021, <https://www.ietf.org/archive/id/draft-claise-opsawg-service-assurance-architecture-05.txt>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Acknowledgements

The authors would like to thank ... for their reviews and feedback.

Authors' Addresses

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

Munish Nayyar
Cisco Systems, Inc.
Milpitas
California,
United States

Email: mnayyar@cisco.com

Adithya Reddy Sesani
Cisco Systems, Inc.
Milpitas
California,
United States

Email: adithyas@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 September 2022

X. He
X. Mao
China Telecom
Q. Ma
X. Zhou
Huawei

5 March 2022

Problem Statement and Use Cases of Adaptive Traffic Data Collection
draft-he-adaptive-collecting-problem-usecases-00

Abstract

IP carrier network needs to provide real-time traffic visibility to help network operators quickly and accurately locate network congestion and packet loss, and make timely path adjustment for deterministic services in order to avoid congestion. It is essential to explore the adaptive traffic data collection mechanism so as to capture real-time network state at minimum resource consumption.

This document summarizes the problems currently faced by network operators when attempting to provide timely traffic data collection to satisfy the various scenarios that require real-time network state and traffic visibility, and aggregates the requirements for adaptive traffic collecting mechanism from a variety of deployment scenarios.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 6 September 2022.

Table of Contents

1. Introduction 2
1.1. Abbreviations 3
2. Terminology 4
3. Problem Statement 4
4. Scenarios of Adaptive Traffic data collection 6
4.1. Multi-dimensional real-time portrait of interface traffic characteristic 6
4.2. Microburst traffic detecting 6
4.3. Congestion avoidance for deterministic services 7
4.4. On-path telemetry based on adaptive traffic sampling . . 7
5. IANA Considerations 8
6. Security Considerations 8
7. References 8
7.1. Normative References 8
7.2. Informative References 9
Authors' Addresses 9

1. Introduction

With the advent of cloud computing, big data and AI, as well as the scale deployment of 5G mobile communication technology, a large number of uRLLC services such as AR/VR, industrial Internet and computing power network have emerged, which puts forward higher requirements for the service quality of IP carrier network. IP carrier network needs to provide real-time traffic visibility to help network operators quickly and accurately locate network congestion and packet loss, and make timely path adjustment for the services of deterministic delay in order to avoid the congested nodes and links. For such business scenarios, the network needs to provide traffic sampling capability in sub seconds or even milliseconds so as to gain real-time network state.

For decades, SNMP and MIBs have been widely deployed and the de facto choice for many monitoring solutions, especially in collecting interface traffic. Arguably the biggest shortcoming of SNMP for those applications concerns the need to rely on periodic polling, because it introduces an additional load on the network and devices,

and it is brittle if polling cycles are missed. Therefore, SNMP has no capability to realize real-time traffic sampling at sub seconds or even milliseconds level. Telemetry, as a revolutionary data acquisition technique, based on pull mechanism that is able to deliver object changes as they happen, overcomes the limitations of SNMP such as "slow speed, low efficiency and more demands for processing capacity". Nevertheless, for the sake of capturing real-time network state, persistent sampling of interface traffic at milliseconds interval will generate a considerable amount of data which may claim too much transport bandwidth and overload the servers for data collection, storage, and analysis. Increasing the data handling capacity is technically feasible but expensive, and difficult to achieve large-scale deployment in operator's networks. It is essential to explore the adaptive traffic data collection mechanism so as to capture real-time network state at minimum resource consumption.

This document summarizes the problems currently faced by network operators when attempting to provide timely traffic data collection to satisfy the various requirements by the aforementioned new scenarios that require real-time network state and traffic visibility. Also, this document aggregates the requirements for adaptive traffic data collection mechanism from a variety of deployment scenarios.

1.1. Abbreviations

AI: Artificial Intelligence

AR: Augmented Reality

VR: Virtual Reality

IP RAN: IP Radio Access Network

DetNet: Deterministic Networking

QoE: Quality of Experience

SLA: Service Level Agreement

uRLLC: ultra Reliable & Low Latency Communication

NMS: Network Management System

IDC: Internet Data Center

SNMP: Simple Network Management Protocol

MIB: Management Information Base

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in this document:

adaptive traffic data collection: Allow servers automatically switch to different telemetry sampling period to collect traffic data according to the threshold change.

3. Problem Statement

As is well known ,IP network, based on statistical multiplexing model, is of traffic burst characteristics. For a long time, operators have obtained traffic visibility from the Network Management System (NMS), and satisfied with 30~40% bandwidth utilization. In spite of such low link usage, many complaints have still been received about poor QoE in delivering applications with the sensitivity of delay and packet loss. The fundamental cause is that the observed average network traffic masks the characteristics of traffic burst, given that SNMP is widely employed in operator's networks to collect network traffic at 5 minutes intervals.

A large quantity of laboratory data and operational data indicate that a microburst phenomenon occurs frequently in operator's carrier networks, such as IP RAN, IP metropolitan network, IP backbone network and IDC. The typical duration of such a microburst is tens to hundreds of milliseconds, easy to cause instantaneous congestion

of the output queue. Network congestion amplifies queuing delay and jitter, it may even give rise to packet loss. All along, solving the problem of network congestion is a major challenge for IP networks. So, the microburst is not beneficial to the deterministic-delay applications. And it is difficult to eliminate the microburst, but must attempt to avoid it.

Although the mechanism of microburst is not very distinct, however, it does not hinder us to detect it. Fortunately, Telemetry (e.g., YANG PUSH [RFC8639] [RFC8641],gNMI [gNMI]) has the capability to collect interface traffic at a higher frequency, i.e., millisecond interval. So, by means of telemetry technique, we can capture the complete aspects of a microburst traffic. However, it is impractical to gain the real-time traffic visibility at the cost of persistent sampling at millisecond intervals. For example, in order to capture a microburst traffic of interface, at least 10-millisecond sampling cycle is necessary. Compared with the today's widely employed 5-minute sampling cycle based on SNMP, the required resources will increase by 30000 times!

It is essential to investigate the adaptive traffic data collection mechanism so as to capture real-time network state at minimum resource consumption. That is to say, in normal non-congested network conditions, which happen at the time of 95% above, minutes-level sampling cycle is enough as it is. But, while detecting a congestion state or congestion trend, sampling period must be timely tuned to milliseconds to capture a microburst traffic of interface. A congestion state or congestion trend of interface is manifested by packet loss due to queue overflow, queue depth beyond the threshold or too high link utilization, which can be defined as Event-triggered data. Such data can be actively pushed through subscription or passively polled through query. Although the microburst phenomenon occurs frequently, it is transient and an on-line detection tool is preferable to find it timely. The traditional method of using CPU on main control board through query is processing resources consuming, the network device must possess built-in hardware designed especially to monitor it.

In order to reduce the excessive consumption of resources caused by millisecond level collection of the single data, batch data such as hundreds of sampled traffic data from an interface can be packaged as a telemetry packet and is sent to the collector. The timestamp is required for every sampled traffic data for the convenience of the collector visualizing the interface traffic trend, And the collector must realize traffic visualization in real-time manner in order that the operators can observe it immediately.

4. Scenarios of Adaptive Traffic data collection

This section presents several typical scenarios which require adaptive traffic data collection to gain real-time network state and traffic visibility at minimum resource consumption.

4.1. Multi-dimensional real-time portrait of interface traffic characteristic

Interface traffic data collection is one of the most important functions for NMS. Today, more and more applications are of latency-sensitive and loss-sensitive characteristic, and the real-time traffic visibility can help operators better understand network performance so as to achieve SLA guarantees. On the other hand, obtaining the holistic and genuine characteristic of interface traffic is also a basic requirement for the statistical multiplexing model of IP network, which is of great significance for traffic prediction, network planning, network capacity expansion, network optimization,etc. However, the traditional NMS based on SNMP has no capability to depict genuine characteristic of interface traffic, and interface traffic data collection based on telemetry techniques is preferable.

It is essential to exploit the adaptive traffic data collection techniques to depict multi-dimensional real-time portrait of interface traffic characteristic at minimum resource consumption. That is to say, in normal non-congested network conditions, which happen at the time of 95% above, minutes-level sampling cycle is enough as it is. But, while detecting a congestion state or congestion trend, sampling cycle must be timely tuned to milliseconds to capture a microburst traffic of interface. Such an adaptive traffic data collection technique can not only reflect the coarse-grained interface traffic characteristics, but also capture the congestion state of interface with finer time granularity. It will be an important tool for the DetNet to obtain real-time network performance. Because of the lower cost, it can be deployed on large-scale in operator’s networks.

4.2. Microburst traffic detecting

Microburst traffic, as an instantaneous congestion phenomenon occurring frequently in IP carrier network, will cause critical delay jitter and even packet loss, which will seriously affect the QoE of latency-sensitive and loss-sensitive applications. The ability of detecting microburst traffic of interface will help network operators quickly and accurately locate network congestion and packet loss, and make timely path adjustment for deterministic-delay services in order to avoid the congested nodes and links.

Because the typical duration of such a microburst is generally tens to hundreds of milliseconds, at least 10-millisecond sampling cycle is necessary. Although the microburst phenomenon occurs frequently, it takes very little time of 24 hours a day. It is not a good approach to observe it through persistent millisecond sampling period. Preferably, we can capture it as soon as a microburst occurs to ensure important diagnose data will not be missed.Because it is transient and an on-line detection tool is required to find it timely. Triggered by the events such as packet loss, queue depth beyond the threshold which is detected timely, sampling period must be timely tuned to milliseconds to capture a microburst traffic of interface. In a word, it is of practical significance to explore the microburst detection technology aiming at minimizing resource consumption.

4.3. Congestion avoidance for deterministic services

Network congestion will rapidly increase queuing delay and jitter, it may even give rise to packet loss, which will seriously affect the QoE of delay-sensitive and packet loss-sensitive applications. The goal of network optimization is to reduce the occurrence of network congestion as much as possible.

It is a complicated problem for network operators to accurately predict the trend of network congestion and make network adjustment in advance. The real-time traffic visibility based on the adaptive traffic data collection techniques can accurately predict the long-term congestion, and quickly capture the instantaneous congestion (i.e,. microburst) of interface. By means of the real-time traffic visibility, the automatic optimization tool (e.g., AI) can make timely path adjustment for key traffic flows. For example, based on the real-time traffic visibility and microburst events (e.g., packet loss, queue depth) collected, the controller can accurately predict the congestion trend of interface and make timely traffic redirection to the non-congested interface for delay deterministic applications.

4.4. On-path telemetry based on adaptive traffic sampling

On-path telemetry is useful for application-aware networking operations. For example, it is critical for the operators who offer high-bandwidth, latency and loss-sensitive services such as video streaming and online gaming to closely monitor the relevant flows in real-time as the basis for any further optimizations. Applying on-path telemetry on all packets of selected flows can still be out of reach. A sampling rate should be set for these flows and only enable telemetry on the sampled packets. However, a too high rate would exhaust the network resource and even cause packet drops; an overly low rate, on the contrary, would result in the loss of information

and inaccuracy of measurements.

An adaptive approach can be used based on the network conditions to dynamically adjust the sampling rate. In normal network state, a low sampling rate is enough to reflect network performance; But, in case of network congestion, the controller is aware of it from the real-time traffic visibility and events data collected (e.g., packet loss, queue depth), and timely adjust the packet sampling rate at very high level. Even all packets of selected flows are applicable so as to acquire real-time measurement data such as latency, jitter and packet loss.

5. IANA Considerations

This document does not include an IANA request.

6. Security Considerations

This document provides an adaptive telemetry mechanism to minimize the resource consumption. The increased complexity of network telemetry may give rise to some security concerns. For example, persistent traffic collection at very high rate (e.g., at millisecond interval) induced by wrong configuration or spurious triggering might exhaust resources of the forwarding plane and the control plane of network device as well as the collector; An inappropriate threshold setting should be avoided. The telemetry data is highly sensitive, which exposes a lot of information about the network and its configuration. Some of that information can make designing attacks against the network much easier (e.g., exact details of what software and patches have been installed), and allows an attacker to determine whether a device may be subject to unprotected security vulnerabilities.

On the other hand, for telemetry interfaces security considerations, NETCONF or gNMI must provide authentication, data integrity,confidentiality, and replay protection. And further study of the security issues will be required.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

7.2. Informative References

[gNMI] "https://github.com/openconfig/gnmi".

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Authors' Addresses

Xiaoming He
China Telecom
Email: hexm4@chinatelecom.cn

Dongfeng Mao
China Telecom
Email: maodf@chinatelecom.cn

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: maqiufang1@huawei.com

Tianran Zhou
Huawei
Email: zhoutianran@huawei.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

YANG Data Types and Groupings for Cryptography
draft-ietf-netconf-crypto-types-22

Abstract

This document presents a YANG 1.1 (RFC 7950) module defining identities, typedefs, and groupings useful to cryptographic applications.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

* AAAA --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 3
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
  1.4. Conventions 5
2. The "ietf-crypto-types" Module 5
  2.1. Data Model Overview 5
  2.2. Example Usage 18
  2.3. YANG Module 27
3. Security Considerations 47
  3.1. No Support for CRMF 48
  3.2. No Support for Key Generation 48
  3.3. Unconstrained Public Key Usage 48
  3.4. Unconstrained Private Key Usage 48
  3.5. Strength of Keys Conveyed 49
  3.6. Encrypting Passwords 49
  3.7. Deletion of Cleartext Key Values 49
  3.8. The "ietf-crypto-types" YANG Module 50
4. IANA Considerations 51
  4.1. The "IETF XML" Registry 51
  4.2. The "YANG Module Names" Registry 52
5. References 52
  5.1. Normative References 52
  5.2. Informative References 53
Appendix A. Change Log 56

A.1. I-D to 00 56
A.2. 00 to 01 56
A.3. 01 to 02 56
A.4. 02 to 03 57
A.5. 03 to 04 58
A.6. 04 to 05 58
A.7. 05 to 06 58
A.8. 06 to 07 58
A.9. 07 to 08 59
A.10. 08 to 09 59
A.11. 09 to 10 59
A.12. 10 to 11 60
A.13. 11 to 12 60
A.14. 12 to 13 60
A.15. 13 to 14 60
A.16. 14 to 15 60
A.17. 15 to 16 61
A.18. 16 to 17 61
A.19. 17 to 18 62
A.20. 18 to 19 62
A.21. 19 to 20 62
A.22. 20 to 21 62
A.23. 21 to 22 62
Acknowledgements 63
Author's Address 63

1. Introduction

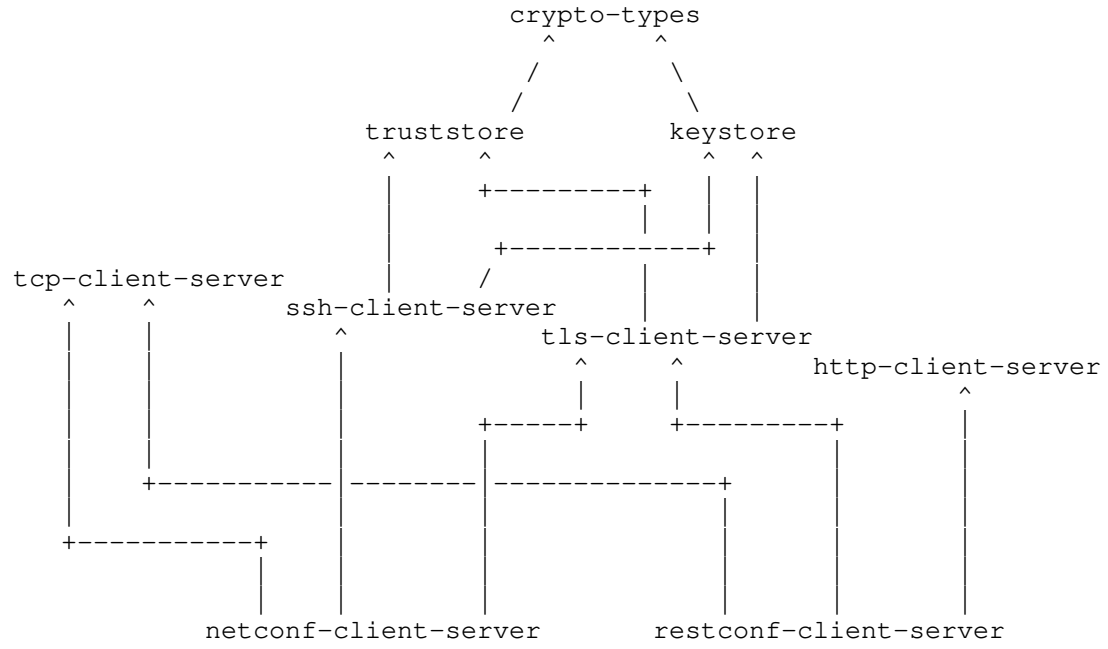
This document presents a YANG 1.1 [RFC7950] module defining identities, typedefs, and groupings useful to cryptographic applications.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-crypto-types" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-crypto-types". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-crypto-types" module in terms of its features, identities, typedefs, and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-crypto-types" module:

```
   Features:
     +-- hidden-keys
     +-- password-encryption
     +-- private-key-encryption
     +-- symmetric-key-encryption
     +-- one-symmetric-key-format
     +-- one-asymmetric-key-format
     +-- cms-encrypted-data-format
     +-- cms-enveloped-data-format
     +-- certificate-expiration-notification
     +-- symmetrically-encrypted-value-format
     +-- asymmetrically-encrypted-value-format
     +-- certificate-signing-request-generation
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-crypto-types" module:

```
   Identities:
     +-- public-key-format
     |  +-- subject-public-key-info-format
     |  +-- ssh-public-key-format
     +-- private-key-format
     |  +-- rsa-private-key-format
     |  +-- ec-private-key-format
     |  +-- one-asymmetric-key-format
     |            {one-asymmetric-key-format}?
     +-- symmetric-key-format
     |  +-- octet-string-key-format
     |  +-- one-symmetric-key-format
     |            {one-symmetric-key-format}?
     +-- encrypted-value-format
        +-- symmetrically-encrypted-value-format
        |  |       {symmetrically-encrypted-value-format}?
        |  +-- cms-encrypted-data-format
        |            {cms-encrypted-data-format}?
        +-- asymmetrically-encrypted-value-format
           |       {asymmetrically-encrypted-value-format}?
           +-- cms-enveloped-data-format
                   {cms-enveloped-data-format}?
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* The diagram shows that there are four base identities. The first three identities are used to indicate the format that key data, while the fourth identity is used to indicate the format for encrypted values. The base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of formats, rather than a specific format.

* The various "leaf" identities define specific encoding formats. The derived identities defined in this document are sufficient for the effort described in Section 1.1 but, by nature of them being identities, additional derived identities MAY be defined by future efforts.

* Identities used to specify uncommon formats are enabled by "feature" statements, allowing applications to support them when needed.

2.1.3. Typedefs

The following diagram illustrates the relationship amongst the "typedef" statements defined in the "ietf-crypto-types" module:

```
Typedefs:
  binary
    +-- csr-info
    +-- csr
    +-- x509
    |  +-- trust-anchor-cert-x509
    |  +-- end-entity-cert-x509
    +-- crl
    +-- ocsp-request
    +-- ocsp-response
    +-- cms
       +-- data-content-cms
       +-- signed-data-cms
       |  +-- trust-anchor-cert-cms
       |  +-- end-entity-cert-cms
       +-- enveloped-data-cms
       +-- digested-data-cms
       +-- encrypted-data-cms
       +-- authenticated-data-cms

    |  The diagram above uses syntax that is similar to but not
    |  defined in [RFC8340].
```

Comments:

* All the typedefs defined in the "ietf-crypto-types" module extend the "binary" type defined in [RFC7950].

* Additionally, all the typedefs define a type for encoding an ASN.1 [ITU.X680.2015] structure using DER [ITU.X690.2015].

* The "trust-anchor-*" and "end-entity-*" typedefs are syntactically identical to their base typedefs and only distinguish themselves by the expected nature of their content. These typedefs are defined to facilitate common modeling needs.

2.1.4. Groupings

The "ietf-crypto-types" module defines the following "grouping" statements:

* encrypted-value-grouping
* password-grouping
* symmetric-key-grouping
* public-key-grouping
* asymmetric-key-pair-grouping
* trust-anchor-cert-grouping
* end-entity-cert-grouping
* generate-csr-grouping
* asymmetric-key-pair-with-cert-grouping
* asymmetric-key-pair-with-certs-grouping

Each of these groupings are presented in the following subsections.

2.1.4.1. The "encrypted-value-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "encrypted-value-grouping" grouping:

```
  grouping encrypted-value-grouping
    +-- encrypted-by
    +-- encrypted-value-format    identityref
    +-- encrypted-value           binary
```

Comments:

* The "encrypted-by" node is an empty container (difficult to see in the diagram) that a consuming module MUST augment key references into. The "ietf-crypto-types" module is unable to populate this container as the module only defines groupings. Section 2.2.1 presents an example illustrating a consuming module populating the "encrypted-by" container.

* The "encrypted-value" node is the value, encrypted by the key referenced by the "encrypted-by" node, and encoded in the format appropriate for the kind of key it was encrypted by.

  - If the value is encrypted by a symmetric key, then the encrypted value is encoded using the format associated with the "symmetrically-encrypted-value-format" identity.

  - If the value is encrypted by an asymmetric key, then the encrypted value is encoded using the format associated with the "asymmetrically-encrypted-value-format" identity.

  See Section 2.1.2 for information about the "format" identities.

2.1.4.2. The "password-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "password-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping password-grouping
    +-- (password-type)
       +--:(cleartext-password)
       |  +-- cleartext-password?   string
       +--:(encrypted-password) {password-encryption}?
          +-- encrypted-password
             +---u encrypted-value-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping password-grouping
    +-- (password-type)
       +--:(cleartext-password)
       |  +-- cleartext-password?   string
       +--:(encrypted-password) {password-encryption}?
          +-- encrypted-password
             +-- encrypted-by
             +-- encrypted-value-format    identityref
             +-- encrypted-value           binary
```

Comments:

* For the referenced grouping statement(s):

  - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "choice" statement enables the password data to be cleartext or encrypted, as follows:

  - The "cleartext-password" node can encode any cleartext value.
  - The "encrypted-password" node's structure is discussed in Section 2.1.4.1.

2.1.4.3. The "symmetric-key-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "symmetric-key-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping symmetric-key-grouping
    +-- key-format?            identityref
    +-- (key-type)
       +--:(cleartext-key)
       |  +-- cleartext-key?   binary
       +--:(hidden-key) {hidden-keys}?
       |  +-- hidden-key?      empty
       +--:(encrypted-key) {symmetric-key-encryption}?
          +-- encrypted-key
             +---u encrypted-value-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping symmetric-key-grouping
    +-- key-format?            identityref
    +-- (key-type)
       +--:(cleartext-key)
       |  +-- cleartext-key?   binary
       +--:(hidden-key) {hidden-keys}?
       |  +-- hidden-key?      empty
       +--:(encrypted-key) {symmetric-key-encryption}?
          +-- encrypted-key
             +-- encrypted-by
             +-- encrypted-value-format    identityref
             +-- encrypted-value           binary
```

Comments:

* For the referenced grouping statement(s):

  - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "key-format" node is an identity-reference to the "symmetric-key-format" abstract base identity discussed in Section 2.1.2, enabling the symmetric key to be encoded using the format defined by any of the derived identities.

* The "choice" statement enables the private key data to be cleartext, encrypted, or hidden, as follows:

  - The "cleartext-key" node can encode any cleartext key value.
  - The "hidden-key" node is of type "empty" as the real value cannot be presented via the management interface.
  - The "encrypted-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.4. The "public-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "public-key-grouping" grouping:

```
grouping public-key-grouping
  +-- public-key-format    identityref
  +-- public-key           binary
```

Comments:

* The "public-key-format" node is an identity-reference to the "public-key-format" abstract base identity discussed in Section 2.1.2, enabling the public key to be encoded using the format defined by any of the derived identities.

* The "public-key" node is the public key data in the selected format. No "choice" statement is used to hide or encrypt the public key data because it is unnecessary to do so for public keys.

2.1.4.5. The "asymmetric-key-pair-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-grouping
    +---u public-key-grouping
    +-- private-key-format?            identityref
    +-- (private-key-type)
       +--:(cleartext-private-key)
       |  +-- cleartext-private-key?   binary
       +--:(hidden-private-key) {hidden-keys}?
       |  +-- hidden-private-key?      empty
       +--:(encrypted-private-key) {private-key-encryption}?
          +-- encrypted-private-key
             +---u encrypted-value-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping asymmetric-key-pair-grouping
    +-- public-key-format              identityref
    +-- public-key                     binary
    +-- private-key-format?            identityref
    +-- (private-key-type)
       +--:(cleartext-private-key)
       |  +-- cleartext-private-key?   binary
       +--:(hidden-private-key) {hidden-keys}?
       |  +-- hidden-private-key?      empty
       +--:(encrypted-private-key) {private-key-encryption}?
          +-- encrypted-private-key
             +-- encrypted-by
             +-- encrypted-value-format    identityref
             +-- encrypted-value           binary
```

Comments:

* For the referenced grouping statement(s):

  - The "public-key-grouping" grouping is discussed in Section 2.1.4.4.
  - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "private-key-format" node is an identity-reference to the "private-key-format" abstract base identity discussed in Section 2.1.2, enabling the private key to be encoded using the format defined by any of the derived identities.

* The "choice" statement enables the private key data to be cleartext, encrypted, or hidden, as follows:

- The "cleartext-private-key" node can encode any cleartext key value.
- The "hidden-private-key" node is of type "empty" as the real value cannot be presented via the management interface.
- The "encrypted-private-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.6. The "certificate-expiration-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "certificate-expiration-grouping" grouping:

```
  grouping certificate-expiration-grouping
    +---n certificate-expiration
            {certificate-expiration-notification}?
       +-- expiration-date    yang:date-and-time
```

Comments:

* This grouping's only purpose is to define the "certificate-expiration" notification statement, used by the groupings defined in Section 2.1.4.7 and Section 2.1.4.8.

* The "certificate-expiration" notification enables servers to notify clients when certificates are nearing expiration.

* The "expiration-date" node indicates when the designated certificate will (or did) expire.

* Identification of the certificate that is expiring is built into the notification itself. For an example, please see Section 2.2.3.

2.1.4.7. The "trust-anchor-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "trust-anchor-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping trust-anchor-cert-grouping
    +-- cert-data?                          trust-anchor-cert-cms
    +---u certificate-expiration-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping trust-anchor-cert-grouping
    +-- cert-data?                trust-anchor-cert-cms
    +---n certificate-expiration
            {certificate-expiration-notification}?
       +-- expiration-date    yang:date-and-time
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.6.

* The "cert-data" node contains a chain of one or more certificates encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.8. The "end-entity-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "end-entity-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping end-entity-cert-grouping
    +-- cert-data?                          end-entity-cert-cms
    +---u certificate-expiration-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping end-entity-cert-grouping
    +-- cert-data?                end-entity-cert-cms
    +---n certificate-expiration
            {certificate-expiration-notification}?
       +-- expiration-date    yang:date-and-time
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.6.

* The "cert-data" node contains a chain of one or more certificates encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.9. The "generate-csr-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "generate-csr-grouping" grouping:

```
  grouping generate-csr-grouping
    +---x generate-certificate-signing-request
            {certificate-signing-request-generation}?
       +---w input
       |  +---w csr-info    ct:csr-info
       +--ro output
          +--ro certificate-signing-request    ct:csr
```

Comments:

* This grouping's only purpose is to define the "generate-certificate-signing-request" action statement, used by the groupings defined in Section 2.1.4.10 and Section 2.1.4.11.

* This action takes as input a "csr-info" type and returns a "csr" type, both of which are discussed in Section 2.1.3.

* For an example, please see Section 2.2.2.

2.1.4.10. The "asymmetric-key-pair-with-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-with-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-with-cert-grouping
    +---u asymmetric-key-pair-grouping
    +---u end-entity-cert-grouping
    +---u generate-csr-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping asymmetric-key-pair-with-cert-grouping
    +-- public-key-format                        identityref
    +-- public-key                               binary
    +-- private-key-format?                      identityref
    +-- (private-key-type)
    |  +--:(cleartext-private-key)
    |  |  +-- cleartext-private-key?             binary
    |  +--:(hidden-private-key) {hidden-keys}?
    |  |  +-- hidden-private-key?                empty
    |  +--:(encrypted-private-key) {private-key-encryption}?
    |     +-- encrypted-private-key
    |        +-- encrypted-by
    |        +-- encrypted-value-format    identityref
    |        +-- encrypted-value           binary
    +-- cert-data?                               end-entity-cert-cms
    +---n certificate-expiration
    |       {certificate-expiration-notification}?
    |  +-- expiration-date    yang:date-and-time
    +---x generate-certificate-signing-request
            {certificate-signing-request-generation}?
       +---w input
       |  +---w csr-info    ct:csr-info
       +--ro output
          +--ro certificate-signing-request    ct:csr
```

Comments:

* This grouping defines an asymmetric key with at most one associated certificate, a commonly needed combination in protocol models.

* For the referenced grouping statement(s):

  - The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.5.
  - The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.8.
  - The "generate-csr-grouping" grouping is discussed in Section 2.1.4.9.

2.1.4.11. The "asymmetric-key-pair-with-certs-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-with-certs-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-with-certs-grouping
    +---u asymmetric-key-pair-grouping
    +-- certificates
    |  +-- certificate* [name]
    |     +-- name?                        string
    |     +---u end-entity-cert-grouping
    +---u generate-csr-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping asymmetric-key-pair-with-certs-grouping
    +-- public-key-format                          identityref
    +-- public-key                                 binary
    +-- private-key-format?                        identityref
    +-- (private-key-type)
    |  +--:(cleartext-private-key)
    |  |  +-- cleartext-private-key?               binary
    |  +--:(hidden-private-key) {hidden-keys}?
    |  |  +-- hidden-private-key?                  empty
    |  +--:(encrypted-private-key) {private-key-encryption}?
    |     +-- encrypted-private-key
    |        +-- encrypted-by
    |        +-- encrypted-value-format    identityref
    |        +-- encrypted-value           binary
    +-- certificates
    |  +-- certificate* [name]
    |     +-- name?                       string
    |     +-- cert-data                   end-entity-cert-cms
    |     +---n certificate-expiration
    |              {certificate-expiration-notification}?
    |        +-- expiration-date    yang:date-and-time
    +---x generate-certificate-signing-request
            {certificate-signing-request-generation}?
       +---w input
       |  +---w csr-info    ct:csr-info
       +--ro output
          +--ro certificate-signing-request    ct:csr
```

Comments:

* This grouping defines an asymmetric key with one or more associated certificates, a commonly needed combination in configuration models.

* For the referenced grouping statement(s):

- The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.5.
- The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.8.
- The "generate-csr-grouping" grouping is discussed in Section 2.1.4.9.

2.1.5. Protocol-accessible Nodes

The "ietf-crypto-types" module does not contain any protocol-accessible nodes, but the module needs to be "implemented", as described in Section 5.6.5 of [RFC7950], in order for the identities in Section 2.1.2 to be defined.

2.2. Example Usage

2.2.1. The "symmetric-key-grouping" and "asymmetric-key-pair-with-certs-grouping" Grouping

The following non-normative module is constructed in order to illustrate the use of the "symmetric-key-grouping" (Section 2.1.4.3), the "asymmetric-key-pair-with-certs-grouping" (Section 2.1.4.11), and the "password-grouping" (Section 2.1.4.2) grouping statements.

Notably, this example illustrates a hidden asymmetric key (ex-hidden-asymmetric-key) has been used to encrypt a symmetric key (ex-encrypted-one-symmetric-based-symmetric-key) that has been used to encrypt another asymmetric key (ex-encrypted-rsa-based-asymmetric-key). Additionally, the symmetric key is also used to encrypt a password (ex-encrypted-password).

```
module ex-crypto-types-usage {
  yang-version 1.1;
  namespace "http://example.com/ns/example-crypto-types-usage";
  prefix ectu;

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

  organization
    "Example Corporation";
  contact
    "YANG Designer <mailto:yang.designer@example.com>";

  description
```

```
       "This module illustrates the 'symmetric-key-grouping'
        and 'asymmetric-key-grouping' groupings defined in
        the 'ietf-crypto-types' module defined in RFC AAAA.";

     revision 2022-03-07 {
       description
         "Initial version";
       reference
         "RFC AAAA: Common YANG Data Types for Cryptography";
     }

     container symmetric-keys {
       description
         "A container of symmetric keys.";
       list symmetric-key {
         key "name";
         description
           "A symmetric key";
         leaf name {
           type string;
           description
             "An arbitrary name for this key.";
         }
         uses ct:symmetric-key-grouping {
           augment "key-type/encrypted-key/encrypted-key/"
                 + "encrypted-by" {
             description
               "Augments in a choice statement enabling the
                encrypting key to be any other symmetric or
                asymmetric key.";
             uses encrypted-by-choice-grouping;
           }
         }
       }
     }
     container asymmetric-keys {
       description
         "A container of asymmetric keys.";
       list asymmetric-key {
         key "name";
         leaf name {
           type string;
           description
             "An arbitrary name for this key.";
         }
         uses ct:asymmetric-key-pair-with-certs-grouping {
           augment "private-key-type/encrypted-private-key/"
                 + "encrypted-private-key/encrypted-by" {
```

```
            description
              "Augments in a choice statement enabling the
               encrypting key to be any other symmetric or
               asymmetric key.";
            uses encrypted-by-choice-grouping;
          }
        }
        description
          "An asymmetric key pair with associated certificates.";
      }
    }
    container passwords {
      description
        "A container of passwords.";
      list password {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this password.";
        }
        uses ct:password-grouping {
          augment "password-type/encrypted-password/"
                + "encrypted-password/encrypted-by" {
            description
              "Augments in a choice statement enabling the
               encrypting key to be any symmetric or
               asymmetric key.";
            uses encrypted-by-choice-grouping;
          }
        }
        description
          "A password.";
      }
    }

    grouping encrypted-by-choice-grouping {
      description
        "A grouping that defines a choice enabling references
         to other keys.";
      choice encrypted-by-choice {
        mandatory true;
        description
          "A choice amongst other symmetric or asymmetric keys.";
        case symmetric-key-ref {
          leaf symmetric-key-ref {
            type leafref {
              path "/ectu:symmetric-keys/ectu:symmetric-key/"
```

```
                + "ectu:name";
            }
            description
              "Identifies the symmetric key that encrypts this key.";
          }
        }
        case asymmetric-key-ref {
          leaf asymmetric-key-ref {
            type leafref {
              path "/ectu:asymmetric-keys/ectu:asymmetric-key/"
                 + "ectu:name";
            }
            description
              "Identifies the asymmetric key that encrypts this key.";
          }
        }
      }
    }
  }
```

The tree diagram [RFC8340] for this example module follows:

```
module: ex-crypto-types-usage
  +--rw symmetric-keys
  |  +--rw symmetric-key* [name]
  |     +--rw name                    string
  |     +--rw key-format?             identityref
  |     +--rw (key-type)
  |        +--:(cleartext-key)
  |        |  +--rw cleartext-key?   binary
  |        +--:(hidden-key) {hidden-keys}?
  |        |  +--rw hidden-key?      empty
  |        +--:(encrypted-key) {symmetric-key-encryption}?
  |           +--rw encrypted-key
  |              +--rw encrypted-by
  |              |  +--rw (encrypted-by-choice)
  |              |     +--:(symmetric-key-ref)
  |              |     |  +--rw symmetric-key-ref?    leafref
  |              |     +--:(asymmetric-key-ref)
  |              |        +--rw asymmetric-key-ref?   leafref
  |              +--rw encrypted-value-format    identityref
  |              +--rw encrypted-value           binary
  +--rw asymmetric-keys
  |  +--rw asymmetric-key* [name]
  |     +--rw name                                  string
  |     +--rw public-key-format                     identityref
  |     +--rw public-key                            binary
  |     +--rw private-key-format?                   identityref
```

```
      |      +--rw (private-key-type)
      |      |  +--:(cleartext-private-key)
      |      |  |  +--rw cleartext-private-key?            binary
      |      |  +--:(hidden-private-key) {hidden-keys}?
      |      |  |  +--rw hidden-private-key?               empty
      |      |  +--:(encrypted-private-key) {private-key-encryption}?
      |      |     +--rw encrypted-private-key
      |      |        +--rw encrypted-by
      |      |        |  +--rw (encrypted-by-choice)
      |      |        |     +--:(symmetric-key-ref)
      |      |        |     |  +--rw symmetric-key-ref?    leafref
      |      |        |     +--:(asymmetric-key-ref)
      |      |        |        +--rw asymmetric-key-ref?   leafref
      |      |        +--rw encrypted-value-format    identityref
      |      |        +--rw encrypted-value           binary
      |      +--rw certificates
      |      |  +--rw certificate* [name]
      |      |     +--rw name                      string
      |      |     +--rw cert-data                 end-entity-cert-cms
      |      |     +---n certificate-expiration
      |      |              {certificate-expiration-notification}?
      |      |        +-- expiration-date    yang:date-and-time
      |      +---x generate-certificate-signing-request
      |                {certificate-signing-request-generation}?
      |        +---w input
      |        |  +---w csr-info    ct:csr-info
      |        +--ro output
      |           +--ro certificate-signing-request    ct:csr
      +--rw passwords
         +--rw password* [name]
            +--rw name                         string
            +--rw (password-type)
               +--:(cleartext-password)
               |  +--rw cleartext-password?   string
               +--:(encrypted-password) {password-encryption}?
                  +--rw encrypted-password
                     +--rw encrypted-by
                     |  +--rw (encrypted-by-choice)
                     |     +--:(symmetric-key-ref)
                     |     |  +--rw symmetric-key-ref?    leafref
                     |     +--:(asymmetric-key-ref)
                     |        +--rw asymmetric-key-ref?   leafref
                     +--rw encrypted-value-format    identityref
                     +--rw encrypted-value           binary
```

Finally, the following example illustrates various symmetric and asymmetric keys as they might appear in configuration:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <symmetric-keys
     xmlns="http://example.com/ns/example-crypto-types-usage"
     xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
     <symmetric-key>
       <name>ex-hidden-symmetric-key</name>
       <hidden-key/>
     </symmetric-key>
     <symmetric-key>
       <name>ex-octet-string-based-symmetric-key</name>
       <key-format>ct:octet-string-key-format</key-format>
       <cleartext-key>BASE64VALUE=</cleartext-key>
     </symmetric-key>
     <symmetric-key>
       <name>ex-one-symmetric-based-symmetric-key</name>
       <key-format>ct:one-symmetric-key-format</key-format>
       <cleartext-key>BASE64VALUE=</cleartext-key>
     </symmetric-key>
     <symmetric-key>
       <name>ex-encrypted-one-symmetric-based-symmetric-key</name>
       <key-format>ct:one-symmetric-key-format</key-format>
       <encrypted-key>
         <encrypted-by>
           <asymmetric-key-ref>ex-hidden-asymmetric-key</asymmetric-key\
   -ref>
         </encrypted-by>
         <encrypted-value-format>
           ct:cms-enveloped-data-format
         </encrypted-value-format>
         <encrypted-value>BASE64VALUE=</encrypted-value>
       </encrypted-key>
     </symmetric-key>
   </symmetric-keys>

   <asymmetric-keys
     xmlns="http://example.com/ns/example-crypto-types-usage"
     xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
     <asymmetric-key>
       <name>ex-hidden-asymmetric-key</name>
       <public-key-format>
         ct:subject-public-key-info-format
       </public-key-format>
       <public-key>BASE64VALUE=</public-key>
       <hidden-private-key/>
       <certificates>
         <certificate>
           <name>ex-hidden-asymmetric-key-cert</name>
```

```
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-rsa-based-asymmetric-key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>
        ct:rsa-private-key-format
      </private-key-format>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      <certificates>
        <certificate>
          <name>ex-cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-one-asymmetric-based-asymmetric-key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>
        ct:one-asymmetric-key-format
      </private-key-format>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-encrypted-rsa-based-asymmetric-key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>
        ct:rsa-private-key-format
      </private-key-format>
      <encrypted-private-key>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
  c-key</symmetric-key-ref>
        </encrypted-by>
        <encrypted-value-format>
          ct:cms-encrypted-data-format
```

```
        </encrypted-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-private-key>
    </asymmetric-key>
  </asymmetric-keys>

  <passwords
    xmlns="http://example.com/ns/example-crypto-types-usage"
    xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <password>
      <name>ex-cleartext-password</name>
      <cleartext-password>super-secret</cleartext-password>
    </password>
    <password>
      <name>ex-encrypted-password</name>
      <encrypted-password>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
        </encrypted-by>
        <encrypted-value-format>
          ct:cms-encrypted-data-format
        </encrypted-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-password>
    </password>
  </passwords>
```

2.2.2. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action, discussed in Section 2.1.4.9, with the NETCONF protocol.

REQUEST

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <asymmetric-keys
      xmlns="http://example.com/ns/example-crypto-types-usage">
      <asymmetric-key>
        <name>ex-key-sect571r1</name>
        <generate-certificate-signing-request>
          <csr-info>BASE64VALUE=</csr-info>
        </generate-certificate-signing-request>
      </asymmetric-key>
    </asymmetric-keys>
  </action>
</rpc>

RESPONSE

<rpc-reply message-id="101"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <certificate-signing-request
     xmlns="http://example.com/ns/example-crypto-types-usage">
     BASE64VALUE=
   </certificate-signing-request>
</rpc-reply>
```

2.2.3. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification, discussed in Section 2.1.4.6, with the NETCONF protocol.

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <asymmetric-keys xmlns="http://example.com/ns/example-crypto-types\
   -usage">
       <asymmetric-key>
         <name>ex-hidden-asymmetric-key</name>
         <certificates>
           <certificate>
             <name>ex-hidden-asymmetric-key</name>
             <certificate-expiration>
               <expiration-date>2018-08-05T14:18:53-05:00</expiration-d\
   ate>
             </certificate-expiration>
           </certificate>
         </certificates>
       </asymmetric-key>
     </asymmetric-keys>
   </notification>
```

2.3. YANG Module

This module has normative references to [RFC2119], [RFC2986], [RFC3447], [RFC4253], [RFC5280], [RFC5652], [RFC5915], [RFC5958], [RFC6031], [RFC6125], [RFC6991], [RFC7093], [RFC8174], [RFC8341], and [ITU.X690.2015].

```
   <CODE BEGINS> file "ietf-crypto-types@2022-03-07.yang"

   module ietf-crypto-types {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-crypto-types";
     prefix ct;

     import ietf-yang-types {
       prefix yang;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     import ietf-netconf-acm {
       prefix nacm;
       reference
         "RFC 8341: Network Configuration Access Control Model";
     }
```

```
  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
     WG List:  NETCONF WG list <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

  description
    "This module defines common YANG types for cryptographic
     applications.

     Copyright (c) 2021 IETF Trust and the persons identified
     as authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Revised
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC AAAA
     (https://www.rfc-editor.org/info/rfcAAAA); see the RFC
     itself for full legal notices.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
     'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
     'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
     are to be interpreted as described in BCP 14 (RFC 2119)
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.";

  revision 2022-03-07 {
    description
      "Initial version";
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

  /****************/
  /*   Features   */
  /****************/

  feature one-symmetric-key-format {
    description
      "Indicates that the server supports the
       'one-symmetric-key-format' identity.";
```

```
  }

  feature one-asymmetric-key-format {
    description
      "Indicates that the server supports the
       'one-asymmetric-key-format' identity.";
  }

  feature symmetrically-encrypted-value-format {
    description
      "Indicates that the server supports the
       'symmetrically-encrypted-value-format' identity.";
  }

  feature asymmetrically-encrypted-value-format {
    description
      "Indicates that the server supports the
       'asymmetrically-encrypted-value-format' identity.";
  }

  feature cms-enveloped-data-format {
    description
      "Indicates that the server supports the
       'cms-enveloped-data-format' identity.";
  }

  feature cms-encrypted-data-format {
    description
      "Indicates that the server supports the
       'cms-encrypted-data-format' identity.";
  }

  feature certificate-signing-request-generation {
    description
      "Indicates that the server implements the
       'generate-certificate-signing-request' action.";
  }

  feature certificate-expiration-notification {
    description
      "Indicates that the server implements the
       'certificate-expiration' notification.";
  }

  feature hidden-keys {
    description
      "Indicates that the server supports hidden keys.";
  }
```

```
    feature password-encryption {
      description
        "Indicates that the server supports password
         encryption.";
    }

    feature symmetric-key-encryption {
      description
        "Indicates that the server supports encryption
         of symmetric keys.";
    }

    feature private-key-encryption {
      description
        "Indicates that the server supports encryption
         of private keys.";
    }

    /************************************************/
    /*   Base Identities for Key Format Structures   */
    /************************************************/

    identity symmetric-key-format {
      description
        "Base key-format identity for symmetric keys.";
    }

    identity public-key-format {
      description
        "Base key-format identity for public keys.";
    }

    identity private-key-format {
      description
        "Base key-format identity for private keys.";
    }

    /************************************************/
    /*   Identities for Private Key Format Structures   */
    /************************************************/

    identity rsa-private-key-format {
      base private-key-format;
      description
        "Indicates that the private key value is encoded
         as an RSAPrivateKey (from RFC 3447).";
      reference
        "RFC 3447: PKCS #1: RSA Cryptography
```

```
                  Specifications Version 2.2";
    }

    identity ec-private-key-format {
      base private-key-format;
      description
        "Indicates that the private key value is encoded
         as an ECPrivateKey (from RFC 5915)";
      reference
        "RFC 5915: Elliptic Curve Private Key Structure";
    }

    identity one-asymmetric-key-format {
      if-feature "one-asymmetric-key-format";
      base private-key-format;
      description
        "Indicates that the private key value is a CMS
         OneAsymmetricKey structure, as defined in RFC 5958,
         encoded using ASN.1 distinguished encoding rules
         (DER), as specified in ITU-T X.690.";
      reference
        "RFC 5958: Asymmetric Key Packages
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }

    /***************************************************/
    /*   Identities for Public Key Format Structures   */
    /***************************************************/

    identity ssh-public-key-format {
      base public-key-format;
      description
        "Indicates that the public key value is an SSH public key,
         as specified by RFC 4253, Section 6.6, i.e.:

           string    certificate or public key format
                     identifier
           byte[n]   key/certificate data.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity subject-public-key-info-format {
      base public-key-format;
```

```
      description
        "Indicates that the public key value is a SubjectPublicKeyInfo
         structure, as described in RFC 5280 encoded using ASN.1
         distinguished encoding rules (DER), as specified in
         ITU-T X.690.";
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }

    /*****************************************************/
    /*   Identities for Symmetric Key Format Structures   */
    /*****************************************************/

    identity octet-string-key-format {
      base symmetric-key-format;
      description
        "Indicates that the key is encoded as a raw octet string.
         The length of the octet string MUST be appropriate for
         the associated algorithm's block size.

         How the associated algorithm is known is outside the
         scope of this module.  This statement also applies when
         the octet string has been encrypted.";
    }

    identity one-symmetric-key-format {
      if-feature "one-symmetric-key-format";
      base symmetric-key-format;
      description
        "Indicates that the private key value is a CMS
         OneSymmetricKey structure, as defined in RFC 6031,
         encoded using ASN.1 distinguished encoding rules
         (DER), as specified in ITU-T X.690.";
      reference
        "RFC 6031: Cryptographic Message Syntax (CMS)
                   Symmetric Key Package Content Type
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
```

```
  }

  /************************************************/
  /*   Identities for Encrypted Value Structures   */
  /************************************************/

  identity encrypted-value-format {
    description
      "Base format identity for encrypted values.";
  }

  identity symmetrically-encrypted-value-format {
    if-feature "symmetrically-encrypted-value-format";
    base encrypted-value-format;
    description
      "Base format identity for symmetrically encrypted
       values.";
  }

  identity asymmetrically-encrypted-value-format {
    if-feature "asymmetrically-encrypted-value-format";
    base encrypted-value-format;
    description
      "Base format identity for asymmetrically encrypted
       values.";
  }

  identity cms-encrypted-data-format {
    if-feature "cms-encrypted-data-format";
    base symmetrically-encrypted-value-format;
    description
      "Indicates that the encrypted value conforms to
       the 'encrypted-data-cms' type with the constraint
       that the 'unprotectedAttrs' value is not set.";
    reference
      "RFC 5652: Cryptographic Message Syntax (CMS)
       ITU-T X.690:
         Information technology - ASN.1 encoding rules:
         Specification of Basic Encoding Rules (BER),
         Canonical Encoding Rules (CER) and Distinguished
         Encoding Rules (DER).";
  }

  identity cms-enveloped-data-format {
    if-feature "cms-enveloped-data-format";
    base asymmetrically-encrypted-value-format;
    description
      "Indicates that the encrypted value conforms to the
```

```
         'enveloped-data-cms' type with the following constraints:

         The EnvelopedData structure MUST have exactly one
         'RecipientInfo'.

         If the asymmetric key supports public key cryptography
         (e.g., RSA), then the 'RecipientInfo' must be a
         'KeyTransRecipientInfo' with the 'RecipientIdentifier'
         using a 'subjectKeyIdentifier' with the value set using
         'method 1' in RFC 7093 over the recipient's public key.

         Otherwise, if the asymmetric key supports key agreement
         (e.g., ECC), then the 'RecipientInfo' must be a
         'KeyAgreeRecipientInfo'.  The 'OriginatorIdentifierOrKey'
         value must use the 'OriginatorPublicKey' alternative.
         The 'UserKeyingMaterial' value must not be present.
         There must be exactly one 'RecipientEncryptedKeys' value
         having the 'KeyAgreeRecipientIdentifier' set to 'rKeyId'
         with the value set using 'method 1' in RFC 7093 over the
         recipient's public key.";
       reference
         "RFC 5652: Cryptographic Message Syntax (CMS)
          RFC 7093:
            Additional Methods for Generating Key
            Identifiers Values
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
     }

     /*************************************************/
     /*   Typedefs for ASN.1 structures from RFC 2986   */
     /*************************************************/

     typedef csr-info {
       type binary;
       description
         "A CertificationRequestInfo structure, as defined in
          RFC 2986, encoded using ASN.1 distinguished encoding
          rules (DER), as specified in ITU-T X.690.";
       reference
         "RFC 2986: PKCS #10: Certification Request Syntax
                    Specification Version 1.7
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
```

```
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
      }

      typedef csr {
        type binary;
        description
          "A CertificationRequest structure, as specified in
           RFC 2986, encoded using ASN.1 distinguished encoding
           rules (DER), as specified in ITU-T X.690.";
        reference
          "RFC 2986:
             PKCS #10: Certification Request Syntax Specification
             Version 1.7
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER).";
      }

      /*************************************************/
      /*   Typedefs for ASN.1 structures from RFC 5280   */
      /*************************************************/

      typedef x509 {
        type binary;
        description
          "A Certificate structure, as specified in RFC 5280,
           encoded using ASN.1 distinguished encoding rules (DER),
           as specified in ITU-T X.690.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER).";
      }

      typedef crl {
        type binary;
        description
          "A CertificateList structure, as specified in RFC 5280,
           encoded using ASN.1 distinguished encoding rules (DER),
           as specified in ITU-T X.690.";
```

```
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }

    /*************************************************/
    /*   Typedefs for ASN.1 structures from RFC 6960   */
    /*************************************************/

    typedef ocsp-request {
      type binary;
      description
        "A OCSPRequest structure, as specified in RFC 6960,
         encoded using ASN.1 distinguished encoding rules
         (DER), as specified in ITU-T X.690.";
      reference
        "RFC 6960:
           X.509 Internet Public Key Infrastructure Online
           Certificate Status Protocol - OCSP
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }

    typedef ocsp-response {
      type binary;
      description
        "A OCSPResponse structure, as specified in RFC 6960,
         encoded using ASN.1 distinguished encoding rules
         (DER), as specified in ITU-T X.690.";
      reference
        "RFC 6960:
           X.509 Internet Public Key Infrastructure Online
           Certificate Status Protocol - OCSP
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }
```

```
     /***********************************************/
     /*   Typedefs for ASN.1 structures from 5652   */
     /***********************************************/

     typedef cms {
       type binary;
       description
         "A ContentInfo structure, as specified in RFC 5652,
          encoded using ASN.1 distinguished encoding rules (DER),
          as specified in ITU-T X.690.";
       reference
         "RFC 5652:
            Cryptographic Message Syntax (CMS)
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
     }

     typedef data-content-cms {
       type cms;
       description
         "A CMS structure whose top-most content type MUST be the
          data content type, as described by Section 4 in RFC 5652.";
       reference
         "RFC 5652: Cryptographic Message Syntax (CMS)";
     }

     typedef signed-data-cms {
       type cms;
       description
         "A CMS structure whose top-most content type MUST be the
          signed-data content type, as described by Section 5 in
          RFC 5652.";
       reference
         "RFC 5652: Cryptographic Message Syntax (CMS)";
     }

     typedef enveloped-data-cms {
       type cms;
       description
         "A CMS structure whose top-most content type MUST be the
          enveloped-data content type, as described by Section 6
          in RFC 5652.";
       reference
         "RFC 5652: Cryptographic Message Syntax (CMS)";
     }
```

```
     typedef digested-data-cms {
       type cms;
       description
         "A CMS structure whose top-most content type MUST be the
          digested-data content type, as described by Section 7
          in RFC 5652.";
       reference
         "RFC 5652: Cryptographic Message Syntax (CMS)";
     }

     typedef encrypted-data-cms {
       type cms;
       description
         "A CMS structure whose top-most content type MUST be the
          encrypted-data content type, as described by Section 8
          in RFC 5652.";
       reference
         "RFC 5652: Cryptographic Message Syntax (CMS)";
     }

     typedef authenticated-data-cms {
       type cms;
       description
         "A CMS structure whose top-most content type MUST be the
          authenticated-data content type, as described by Section 9
          in RFC 5652.";
       reference
         "RFC 5652: Cryptographic Message Syntax (CMS)";
     }

     /*********************************************************/
     /*   Typedefs for ASN.1 structures related to RFC 5280   */
     /*********************************************************/

     typedef trust-anchor-cert-x509 {
       type x509;
       description
         "A Certificate structure that MUST encode a self-signed
          root certificate.";
     }

     typedef end-entity-cert-x509 {
       type x509;
       description
         "A Certificate structure that MUST encode a certificate
          that is neither self-signed nor having Basic constraint
          CA true.";
     }
```

```
     /*********************************************************/
     /*   Typedefs for ASN.1 structures related to RFC 5652   */
     /*********************************************************/

     typedef trust-anchor-cert-cms {
       type signed-data-cms;
       description
         "A CMS SignedData structure that MUST contain the chain of
          X.509 certificates needed to authenticate the certificate
          presented by a client or end-entity.

          The CMS MUST contain only a single chain of certificates.
          The client or end-entity certificate MUST only authenticate
          to last intermediate CA certificate listed in the chain.

          In all cases, the chain MUST include a self-signed root
          certificate.  In the case where the root certificate is
          itself the issuer of the client or end-entity certificate,
          only one certificate is present.

          This CMS structure MAY (as applicable where this type is
          used) also contain suitably fresh (as defined by local
          policy) revocation objects with which the device can
          verify the revocation status of the certificates.

          This CMS encodes the degenerate form of the SignedData
          structure that is commonly used to disseminate X.509
          certificates and revocation objects (RFC 5280).";
       reference
         "RFC 5280:
            Internet X.509 Public Key Infrastructure Certificate
            and Certificate Revocation List (CRL) Profile.";
     }

     typedef end-entity-cert-cms {
       type signed-data-cms;
       description
         "A CMS SignedData structure that MUST contain the end
          entity certificate itself, and MAY contain any number
          of intermediate certificates leading up to a trust
          anchor certificate.  The trust anchor certificate
          MAY be included as well.

          The CMS MUST contain a single end entity certificate.
          The CMS MUST NOT contain any spurious certificates.

          This CMS structure MAY (as applicable where this type is
          used) also contain suitably fresh (as defined by local
```

```
          policy) revocation objects with which the device can
          verify the revocation status of the certificates.

          This CMS encodes the degenerate form of the SignedData
          structure that is commonly used to disseminate X.509
          certificates and revocation objects (RFC 5280).";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile.";
      }

      /*****************/
      /*   Groupings   */
      /*****************/

      grouping encrypted-value-grouping {
        description
          "A reusable grouping for a value that has been encrypted by
           a referenced symmetric or asymmetric key.";
        container encrypted-by {
          nacm:default-deny-write;
          description
            "An empty container enabling a reference to the key that
             encrypted the value to be augmented in.  The referenced
             key MUST be a symmetric key or an asymmetric key.

             A symmetric key MUST be referenced via a leaf node called
             'symmetric-key-ref'.  An asymmetric key MUST be referenced
             via a leaf node called 'asymmetric-key-ref'.

             The leaf nodes MUST be direct descendants in the data tree,
             and MAY be direct descendants in the schema tree.";
        }
        leaf encrypted-value-format {
          type identityref {
            base encrypted-value-format;
          }
          mandatory true;
          description
            "Identifies the format of the 'encrypted-value' leaf.

             If 'encrypted-by' points to a symmetric key, then a
             'symmetrically-encrypted-value-format' based identity
             MUST by set (e.g., cms-encrypted-data-format).

             If 'encrypted-by' points to an asymmetric key, then an
             'asymmetrically-encrypted-value-format' based identity
```

```
            MUST by set (e.g., cms-enveloped-data-format).";
      }
      leaf encrypted-value {
        nacm:default-deny-write;
        type binary;
        must '../encrypted-by';
        mandatory true;
        description
          "The value, encrypted using the referenced symmetric
           or asymmetric key.  The value MUST be encoded using
           the format associated with the 'encrypted-value-format'
           leaf.";
      }
    }

    grouping password-grouping {
      description
        "A password that MAY be encrypted.";
      choice password-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between password types.";
        case cleartext-password {
          leaf cleartext-password {
            nacm:default-deny-all;
            type string;
            description
              "The cleartext value of the password.";
          }
        }
        case encrypted-password {
          if-feature "password-encryption";
          container encrypted-password {
            description
              "A container for the encrypted password value.";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping symmetric-key-grouping {
      description
        "A symmetric key.";
      leaf key-format {
        nacm:default-deny-write;
        type identityref {
```

```
          base symmetric-key-format;
        }
        description
          "Identifies the symmetric key's format.  Implementations
           SHOULD ensure that the incoming symmetric key value is
           encoded in the specified format.

           For encrypted keys, the value is the same as it would
           have been if the key were not encrypted.";
      }
      choice key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-key {
          leaf cleartext-key {
            nacm:default-deny-all;
            type binary;
            must '../key-format';
            description
              "The binary value of the key.  The interpretation of
               the value is defined by the 'key-format' field.";
          }
        }
        case hidden-key {
          if-feature "hidden-keys";
          leaf hidden-key {
            type empty;
            must 'not(../key-format)';
            description
              "A hidden key.  How such keys are created is outside
               the scope of this module.";
          }
        }
        case encrypted-key {
          if-feature "symmetric-key-encryption";
          container encrypted-key {
            must '../key-format';
            description
              "A container for the encrypted symmetric key value.
               The interpretation of the 'encrypted-value' node
               is via the 'key-format' node";
            uses encrypted-value-grouping;
          }
        }
      }
    }
```

```
    grouping public-key-grouping {
      description
        "A public key.";
      leaf public-key-format {
        nacm:default-deny-write;
        type identityref {
          base public-key-format;
        }
        mandatory true;
        description
          "Identifies the public key's format. Implementations SHOULD
           ensure that the incoming public key value is encoded in the
           specified format.";
      }
      leaf public-key {
        nacm:default-deny-write;
        type binary;
        mandatory true;
        description
          "The binary value of the public key.  The interpretation
           of the value is defined by 'public-key-format' field.";
      }
    }

    grouping asymmetric-key-pair-grouping {
      description
        "A private key and its associated public key.  Implementations
         SHOULD ensure that the two keys are a matching pair.";
      uses public-key-grouping;
      leaf private-key-format {
        nacm:default-deny-write;
        type identityref {
          base private-key-format;
        }
        description
          "Identifies the private key's format.  Implementations SHOULD
           ensure that the incoming private key value is encoded in the
           specified format.

           For encrypted keys, the value is the same as it would have
           been if the key were not encrypted.";
      }
      choice private-key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-private-key {
```

```
          leaf cleartext-private-key {
            nacm:default-deny-all;
            type binary;
            must '../private-key-format';
            description
              "The value of the binary key  The key's value is
               interpreted by the 'private-key-format' field.";
          }
        }
        case hidden-private-key {
          if-feature "hidden-keys";
          leaf hidden-private-key {
            type empty;
            must 'not(../private-key-format)';
            description
              "A hidden key.  How such keys are created is
               outside the scope of this module.";
          }
        }
        case encrypted-private-key {
          if-feature "private-key-encryption";
          container encrypted-private-key {
            must '../private-key-format';
            description
              "A container for the encrypted asymmetric private key
               value.  The interpretation of the 'encrypted-value'
               node is via the 'private-key-format' node";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping certificate-expiration-grouping {
      description
        "A notification for when a certificate is about to, or
         already has, expired.";
      notification certificate-expiration {
        if-feature "certificate-expiration-notification";
        description
          "A notification indicating that the configured certificate
           is either about to expire or has already expired.  When to
           send notifications is an implementation specific decision,
           but it is RECOMMENDED that a notification be sent once a
           month for 3 months, then once a week for four weeks, and
           then once a day thereafter until the issue is resolved.";
        leaf expiration-date {
          type yang:date-and-time;
```

```
          mandatory true;
          description
            "Identifies the expiration date on the certificate.";
        }
      }
    }

    grouping trust-anchor-cert-grouping {
      description
        "A trust anchor certificate, and a notification for when
         it is about to (or already has) expire.";
      leaf cert-data {
        nacm:default-deny-write;
        type trust-anchor-cert-cms;
        description
          "The binary certificate data for this certificate.";
      }
      uses certificate-expiration-grouping;
    }

    grouping end-entity-cert-grouping {
      description
        "An end entity certificate, and a notification for when
         it is about to (or already has) expire.  Implementations
         SHOULD assert that, where used, the end entity certificate
         contains the expected public key.";
      leaf cert-data {
        nacm:default-deny-write;
        type end-entity-cert-cms;
        description
          "The binary certificate data for this certificate.";
      }
      uses certificate-expiration-grouping;
    }

    grouping generate-csr-grouping {
      description
        "Defines the 'generate-certificate-signing-request' action.";
      action generate-certificate-signing-request {
        if-feature "certificate-signing-request-generation";
        nacm:default-deny-all;
        description
          "Generates a certificate signing request structure for
           the associated asymmetric key using the passed subject
           and attribute values.

           This action statement is only available when the
           associated 'public-key-format' node's value is
```

```
           'subject-public-key-info-format'.";
        reference
          "RFC 6125:
           Representation and Verification of Domain-Based
           Application Service Identity within Internet Public Key
           Infrastructure Using X.509 (PKIX) Certificates in the
           Context of Transport Layer Security (TLS)";
        input {
          leaf csr-info {
            type ct:csr-info;
            mandatory true;
            description
              "A CertificationRequestInfo structure, as defined in
               RFC 2986.

               Enables the client to provide a fully-populated
               CertificationRequestInfo structure that the server
               only needs to sign in order to generate the complete
               'CertificationRequest' structure to return in the
               'output'.

               The 'AlgorithmIdentifier' field contained inside
               the 'SubjectPublicKeyInfo' field MUST be one known
               to be supported by the device.";
            reference
              "RFC 2986:
                 PKCS #10: Certification Request Syntax Specification
               RFC AAAA:
                 YANG Data Types and Groupings for Cryptography";
          }
        }
        output {
          leaf certificate-signing-request {
            type ct:csr;
            mandatory true;
            description
              "A CertificationRequest structure, as defined in
               RFC 2986.";
            reference
              "RFC 2986:
                 PKCS #10: Certification Request Syntax Specification
               RFC AAAA:
                 YANG Data Types and Groupings for Cryptography";
          }
        }
      }
    } // generate-csr-grouping
```

```
    grouping asymmetric-key-pair-with-cert-grouping {
      description
        "A private/public key pair and an associated certificate.
         Implementations SHOULD assert that certificates contain
         the matching public key.";
      uses asymmetric-key-pair-grouping;
      uses end-entity-cert-grouping;
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-cert-grouping

    grouping asymmetric-key-pair-with-certs-grouping {
      description
        "A private/public key pair and associated certificates.
         Implementations SHOULD assert that certificates contain
         the matching public key.";
      uses asymmetric-key-pair-grouping;
      container certificates {
        nacm:default-deny-write;
        description
          "Certificates associated with this asymmetric key.";
        list certificate {
          key "name";
          description
            "A certificate for this asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the certificate.";
          }
          uses end-entity-cert-grouping {
            refine "cert-data" {
              mandatory true;
            }
          }
        }
      }
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-certs-grouping

  }

  <CODE ENDS>
```

3. Security Considerations

3.1. No Support for CRMF

This document uses PKCS #10 [RFC2986] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [RFC4211] was considered, but it was unclear if there was market demand for it. If it is desired to support CRMF in the future, a backwards compatible solution can be defined at that time.

3.2. No Support for Key Generation

Early revisions of this document included "rpc" statements for generating symmetric and asymmetric keys. These statements were removed due to an inability to obtain consensus for how to identify the key-algorithm to use. Thusly, the solution presented in this document only supports keys to be configured via an external client, which does not support Security best practice.

3.3. Unconstrained Public Key Usage

This module defines the "public-key-grouping" grouping, which enables the configuration of public keys without constraints on their usage, e.g., what operations the key is allowed to be used for (encryption, verification, both).

The "asymmetric-key-pair-grouping" grouping uses the aforementioned "public-key-grouping" grouping, and carries the same traits.

The "asymmetric-key-pair-with-cert-grouping" grouping uses the aforementioned "asymmetric-key-pair-grouping" grouping, whereby each certificate may constrain the usage of the public key according to local policy.

3.4. Unconstrained Private Key Usage

This module defines the "asymmetric-key-pair-grouping" grouping, which enables the configuration of private keys without constraints on their usage, e.g., what operations the key is allowed to be used for (e.g., signature, decryption, both).

The "asymmetric-key-pair-with-cert-grouping" uses the aforementioned "asymmetric-key-pair-grouping" grouping, whereby configured certificates (e.g., identity certificates) may constrain the use of the public key according to local policy.

3.5. Strength of Keys Conveyed

When accessing key values, it is desireable that implementations ensure that the strength of the keys being accessed is not greater than the strength of the underlying secure transport connection over which the keys are conveyed. However, comparing key strengths can be complicated and difficult to implement in practice.

That said, expert Security opinion suggests that already it is infeasible to break a 128-bit symmetric key using a classical computer, and thus the concern for conveying higher-strength keys begins to lose its allure.

Implementations SHOULD only use secure transport protocols meeting local policy. A reasonable policy may, e.g., state that only ciphersuites listed as "recommended" by the IETF be used (e.g., [RFC7525] for TLS).

3.6. Encrypting Passwords

The module contained within this document enables passwords to be encrypted. Passwords may be encrypted via a symmetric key using the "cms-encrypted-data-format" format. This format uses the CMS EncryptedData structure, which allows any encryption algorithm to be used.

In order to thwart rainbow attacks, algorithms that result in a unique output for the same input SHOULD NOT be used. For instance, AES using "ECB" SHOULD NOT be used to encrypt passwords, whereas "CBC" mode is permissible since an unpredictable initialization vector (IV) MUST be used for each use.

3.7. Deletion of Cleartext Key Values

This module defines storage for cleartext key values that SHOULD be zeroized when deleted, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

The cleartext key values are the "cleartext-key" node defined in the "symmetric-key-grouping" grouping (Section 2.1.4.3) and the "cleartext-private-key" node defined in the "asymmetric-key-pair-grouping" grouping ("Section 2.1.4.5).

3.8. The "ietf-crypto-types" YANG Module

The YANG module in this document defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only defines groupings, these considerations are primarily for the designers of other modules that use these groupings.

Some of the readable data nodes defined in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

* The "cleartext-key" node:

  The "cleartext-key" node defined in the "symmetric-key-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

* The "cleartext-private-key" node:

  The "cleartext-private-key" node defined in the "asymmetric-key-pair-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied.

* The "cert-data" node:

The "cert-data" node, defined in both the "trust-anchor-cert-grouping" and "end-entity-cert-grouping" groupings, is additionally sensitive to read operations, as certificates sometimes convey personally identifying information (especially end-entity certificates). However, as it is commonly understood that certificates are "public", the NACM extension "nacm:default-deny-write" (not "default-deny-all") has been applied. It is RECOMMENDED that implementations adjust read-access to certificates to comply with local policy.

All the writable data nodes defined by all the groupings defined in this module may be considered sensitive or vulnerable in some network environments. For instance, even the modification of a public key or a certificate can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been applied to all the data nodes defined in the module.

Some of the operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

* generate-certificate-signing-request:

  This "action" statement SHOULD only be executed by authorized users. For this reason, the NACM extension "default-deny-all" has been applied. Note that NACM uses "default-deny-all" to protect "RPC" and "action" statements; it does not define, e.g., an extension called "default-deny-execute".

  For this action, it is RECOMMENDED that implementations assert channel binding [RFC5056], so as to ensure that the application layer that sent the request is the same as the device authenticated when the secure transport layer was established.

4. IANA Considerations

4.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the "IETF XML" registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-crypto-types
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

4.2. The "YANG Module Names" Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration is requested:

```
name:         ietf-crypto-types
namespace:    urn:ietf:params:xml:ns:yang:ietf-crypto-types
prefix:       ct
reference:    RFC AAAA
```

5. References

5.1. Normative References

[ITU.X680.2015]
International Telecommunication Union, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2015, August 2015, <https://www.itu.int/rec/T-REC-X.680/>.

[ITU.X690.2015]
International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2015, August 2015, <https://www.itu.int/rec/T-REC-X.690/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <https://www.rfc-editor.org/info/rfc3447>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC6031] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Symmetric Key Package Content Type", RFC 6031, DOI 10.17487/RFC6031, December 2010, <https://www.rfc-editor.org/info/rfc6031>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7093] Turner, S., Kent, S., and J. Manger, "Additional Methods for Generating Key Identifiers Values", RFC 7093, DOI 10.17487/RFC7093, December 2013, <https://www.rfc-editor.org/info/rfc7093>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

5.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <https://www.rfc-editor.org/info/rfc5056>.

[RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", RFC 5915, DOI 10.17487/RFC5915, June 2010, <https://www.rfc-editor.org/info/rfc5915>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. I-D to 00

* Removed groupings and notifications.
* Added typedefs for identityrefs.
* Added typedefs for other RFC 5280 structures.
* Added typedefs for other RFC 5652 structures.
* Added convenience typedefs for RFC 4253, RFC 5280, and RFC 5652.

A.2. 00 to 01

* Moved groupings from the draft-ietf-netconf-keystore here.

A.3. 01 to 02

* Removed unwanted "mandatory" and "must" statements.

* Added many new crypto algorithms (thanks Haiguang!)

* Clarified in asymmetric-key-pair-with-certs-grouping, in certificates/certificate/name/description, that if the name MUST NOT match the name of a certificate that exists independently in <operational>, enabling certs installed by the manufacturer (e.g., an IDevID).

A.4. 02 to 03

* renamed base identity 'asymmetric-key-encryption-algorithm' to 'asymmetric-key-algorithm'.

* added new 'asymmetric-key-algorithm' identities for secp192r1, secp224r1, secp256r1, secp384r1, and secp521r1.

* removed 'mac-algorithm' identities for mac-aes-128-ccm, mac-aes-192-ccm, mac-aes-256-ccm, mac-aes-128-gcm, mac-aes-192-gcm, mac-aes-256-gcm, and mac-chacha20-poly1305.

* for all -cbc and -ctr identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-algorithm'.

* for all -ccm and -gcm identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-and-mac-algorithm' and renamed the identity to remove the "enc-" prefix.

* for all the 'signature-algorithm' based identities, renamed from 'rsa-*' to 'rsassa-*'.

* removed all of the "x509v3-" prefixed 'signature-algorithm' based identities.

* added 'key-exchange-algorithm' based identities for 'rsaes-oaep' and 'rsaes-pkcs1-v1_5'.

* renamed typedef 'symmetric-key-encryption-algorithm-ref' to 'symmetric-key-algorithm-ref'.

* renamed typedef 'asymmetric-key-encryption-algorithm-ref' to 'asymmetric-key-algorithm-ref'.

* added typedef 'encryption-and-mac-algorithm-ref'.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.5. 03 to 04

* ran YANG module through formatter.

A.6. 04 to 05

* fixed broken symlink causing reformatted YANG module to not show.

A.7. 05 to 06

* Added NACM annotations.

* Updated Security Considerations section.

* Added 'asymmetric-key-pair-with-cert-grouping' grouping.

* Removed text from 'permanently-hidden' enum regarding such keys not being backed up or restored.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Added an explanation to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements as for why the nodes are not mandatory (e.g., because they may exist only in <operational>.

* Added 'must' expressions to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements ensuring sibling nodes are either all exist or do not all exist.

* Added an explanation to the 'permanently-hidden' that the value cannot be configured directly by clients and servers MUST fail any attempt to do so.

* Added 'trust-anchor-certs-grouping' and 'end-entity-certs-grouping' (the plural form of existing groupings).

* Now states that keys created in <operational> by the *-hidden-key actions are bound to the lifetime of the parent 'config true' node, and that subsequent invocations of either action results in a failure.

A.8. 06 to 07

* Added clarifications that implementations SHOULD assert that configured certificates contain the matching public key.

* Replaced the 'generate-hidden-key' and 'install-hidden-key' actions with special 'crypt-hash' -like input/output values.

A.9. 07 to 08

* Removed the 'generate-key and 'hidden-key' features.

* Added grouping symmetric-key-grouping

* Modified 'asymmetric-key-pair-grouping' to have a 'choice' statement for the keystone module to augment into, as well as replacing the 'union' with leafs (having different NACM settings.

A.10. 08 to 09

* Converting algorithm from identities to enumerations.

A.11. 09 to 10

* All the below changes are to the algorithm enumerations defined in ietf-crypto-types.

* Add in support for key exchange over x.25519 and x.448 based on RFC 8418.

* Add in SHAKE-128, SHAKE-224, SHAKE-256, SHAKE-384 and SHAKE 512

* Revise/add in enum of signature algorithm for x25519 and x448

* Add in des3-cbc-sha1 for IPSec

* Add in sha1-des3-kd for IPSec

* Add in definit for rc4-hmac and rc4-hmac-exp. These two algorithms have been deprecated in RFC 8429. But some existing draft in i2nsf may still want to use them.

* Add x25519 and x448 curve for asymmetric algorithms

* Add signature algorithms ed25519, ed25519-cts, ed25519ph

* add signature algorithms ed448, ed448ph

* Add in rsa-sha2-256 and rsa-sha2-512 for SSH protocols (rfc8332)

A.12. 10 to 11

* Added a "key-format" identity.

* Added symmetric keys to the example in Section 2.2.

A.13. 11 to 12

* Removed all non-essential (to NC/RC) algorithm types.

* Moved remaining algorithm types each into its own module.

* Added a 'config false' "algorithms-supported" list to each of the algorithm-type modules.

A.14. 12 to 13

* Added the four features: "[encrypted-]one-[a]symmetric-key-format", each protecting a 'key-format' identity of the same name.

* Added 'must' expressions asserting that the 'key-format' leaf exists whenever a non-hidden key is specified.

* Improved the 'description' statements and added 'reference' statements for the 'key-format' identities.

* Added a questionable forward reference to "encrypted-*" leafs in a couple 'when' expressions.

* Did NOT move "config false" alg-supported lists to SSH/TLS drafts.

A.15. 13 to 14

* Resolved the "FIXME: forward ref" issue by modulating 'must', 'when', and 'mandatory' expressions.

* Moved the 'generatesymmetric-key' and 'generate-asymmetric-key' actions from ietf-keystore to ietf-crypto-types, now as RPCs.

* Cleaned up various description statements and removed lingering FIXMEs.

* Converted the "iana-<alg-type>-algs" YANG modules to IANA registries with instructions for how to generate modules from the registries, whenever they may be updated.

A.16. 14 to 15

* Removed the IANA-maintained registries for symmetric, asymmetric, and hash algorithms.

* Removed the "generate-symmetric-key" and "generate-asymmetric-key" RPCs.

* Removed the "algorithm" node in the various symmetric and asymmetric key groupings.

* Added 'typedef csr' and 'feature certificate-signing-request-generation'.

* Refined a usage of "end-entity-cert-grouping" to make the "cert" node mandatory true.

* Added a "Note to Reviewers" note to first page.

A.17. 15 to 16

* Updated draft title (refer to "Groupings" too).

* Removed 'end-entity-certs-grouping' as it wasn't being used anywhere.

* Removed 'trust-anchor-certs-grouping' as it was no longer being used after modifying 'local-or-truststore-certs-grouping' to use lists (not leaf-lists).

* Renamed "cert" to "cert-data" in trust-anchor-cert-grouping.

* Added "csr-info" typedef, to complement the existing "csr" typedef.

* Added "ocsp-request" and "ocsp-response" typedefs, to complement the existing "crl" typedef.

* Added "encrypted" cases to both symmetric-key-grouping and asymmetric-key-pair-grouping (Moved from Keystore draft).

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.18. 16 to 17

* [Re]-added a "Strength of Keys Configured" Security Consideration

* Prefixed "cleartext-" in the "key" and "private-key" node names.

A.19. 17 to 18

* Fixed issues found by the SecDir review of the "keystore" draft.

* Added "password-grouping", discussed during the IETF 108 session.

A.20. 18 to 19

* Added a "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.

* Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.

* Changed the encryption strategy, after conferring with Russ Housley.

* Added a "password-grouping" example to the "crypto-types-usage" example.

* Added an "Encrypting Passwords" section to Security Consideration.

* Addressed other comments raised by YANG Doctor.

A.21. 19 to 20

* Nits found via YANG Doctors reviews.

* Aligned modules with `pyang -f` formatting.

A.22. 20 to 21

* Replaced "base64encodedvalue==" with "BASE64VALUE=".

* Accommodated SecDir review by Valery Smyslov.

A.23. 21 to 22

* fixup the 'WG Web' and 'WG List' lines in YANG module(s)

* fixup copyright (i.e., s/Simplified/Revised/) in YANG module(s)

* added 'hidden-keys' feature.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Balazs Kovacs, Eric Voit, Juergen Schoenwaelder, Liang Xia, Martin Bjoerklund, Nick Hancock, Rich Salz, Rob Wilton, Russ Housley, Sandra Murphy, Tom Petch, Valery Smyslov, and Wang Haiguang.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

YANG Groupings for HTTP Clients and HTTP Servers
draft-ietf-netconf-http-client-server-09

Abstract

This document defines two YANG modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based protocols (not for complete web servers or browsers).

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements (note: not all may be present):

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types

* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server

* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server

* GGGG --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Relation to other RFCs 3
1.2. Specification Language 5
1.3. Adherence to the NMDA 5
1.4. Conventions 5
2. The "ietf-http-client" Module 5
2.1. Data Model Overview 5
2.2. Example Usage 8
2.3. YANG Module 10
3. The "ietf-http-server" Module 16
3.1. Data Model Overview 16
3.2. Example Usage 18
3.3. YANG Module 19
4. Security Considerations 24
4.1. The "ietf-http-client" YANG Module 24

4.2. The "ietf-http-server" YANG Module 25
5. IANA Considerations 26
5.1. The "IETF XML" Registry 26
5.2. The "YANG Module Names" Registry 26
6. References 26
6.1. Normative References 26
6.2. Informative References 27
Appendix A. Change Log 29
A.1. 00 to 01 29
A.2. 01 to 02 29
A.3. 02 to 03 29
A.4. 03 to 04 30
A.5. 04 to 05 30
A.6. 05 to 06 30
A.7. 06 to 07 30
A.8. 07 to 08 31
A.9. 08 to 09 31
Acknowledgements 31
Author's Address 31

1. Introduction

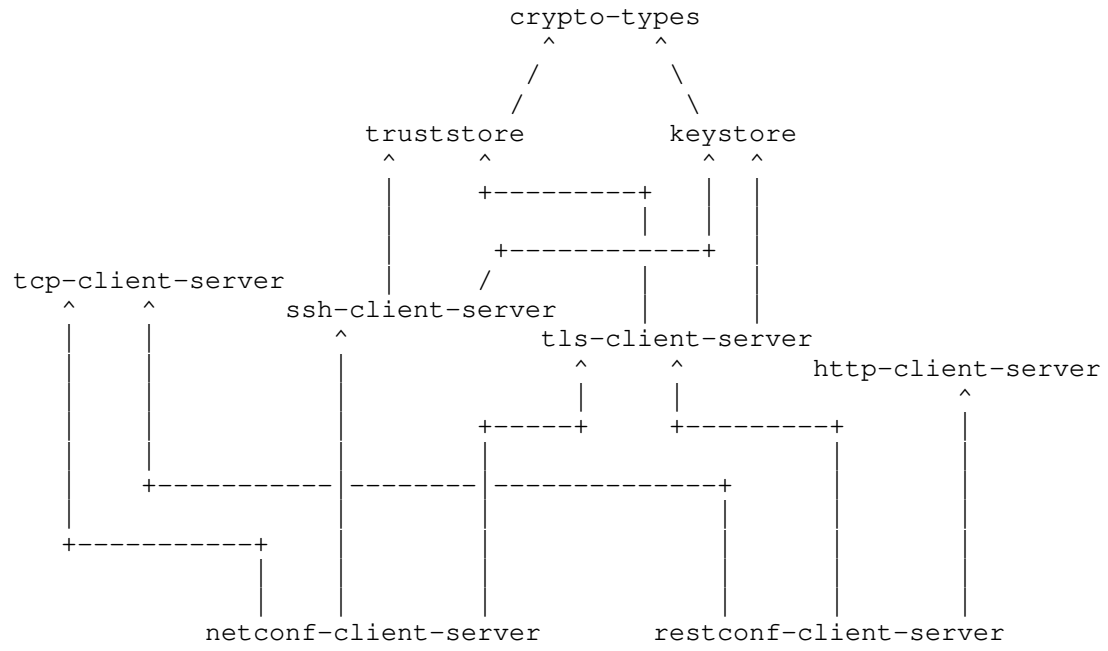
This document defines two YANG 1.1 [RFC7950] modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based protocols (not for complete web servers or browsers).

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-http-client" Module

This section defines a YANG 1.1 module called "ietf-http-client". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-http-client" module in terms of its features and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-http-client" module:

```
Features:
  +-- proxy-connect
  +-- basic-auth
  +-- tcp-supported
  +-- tls-supported
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Groupings

The "ietf-http-client" module defines the following "grouping" statements:

* http-client-identity-grouping
* http-client-grouping
* http-client-stack-grouping

Each of these groupings are presented in the following subsections.

2.1.2.1. The "http-client-identity-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-client-identity-grouping" grouping:

```
  grouping http-client-identity-grouping
    +-- client-identity!
       +-- (auth-type)
          +--:(basic)
             +-- basic {basic-auth}?
                +-- user-id                 string
                +---u ct:password-grouping
```

Comments:

* This grouping exists because it is used three times by the "http-client-grouping" discussed in Section 2.1.2.2.

* The "client-identity" node is a "presence" container so the mandatory descendant nodes do not imply that this node must be configured, as a client identity may be configured at protocol layers.

* The "basic" authentication scheme is the only scheme defined by this module, albeit it must be enabled via the "basic-auth" feature (see Section 2.1.1).

* Other authentication schemes MAY be augmented in as needed by the application.

2.1.2.2. The "http-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-client-grouping" grouping:

```
  grouping http-client-grouping
    +---u http-client-identity-grouping
    +-- proxy-connect! {proxy-connect}?
       +-- (proxy-type)
          +--:(http)
          |  +-- http-proxy
          |     +-- tcp-client-parameters
          |     |  +---u tcpc:tcp-client-grouping
          |     +-- http-client-parameters
          |        +---u http-client-identity-grouping
          +--:(https)
             +-- https-proxy
                +-- tcp-client-parameters
                |  +---u tcpc:tcp-client-grouping
                +-- tls-client-parameters
                |  +---u tlsc:tls-client-grouping
                +-- http-client-parameters
                   +---u http-client-identity-grouping
```

Comments:

* The "http-client-grouping" defines the configuration for just "HTTP" part of a protocol stack. It does not, for instance, define any configuration for the "TCP" or "TLS" protocol layers (for that, see Section 2.1.2.3).

* Beyond configuring the client's identity, via the "http-client-identity-grouping" grouping discussed in Section 2.1.2.1, this grouping defines support for HTTP-proxies, albeit it must be enabled via a "feature" statement.

* The "proxy-connect" node is a "presence" container so the mandatory descendant nodes do not imply that this node must be configured, assuming the server supports the "proxy-connect" feature.

* For the referenced grouping statement(s):

  - The "http-client-identity-grouping" grouping is discussed in Section 2.1.2.1.
  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].

2.1.2.3. The "http-client-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-client-stack-grouping" grouping:

```
  grouping http-client-stack-grouping
    +-- (transport)
       +--:(tcp) {tcp-supported}?
       |  +-- tcp
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- http-client-parameters
       |        +---u http-client-grouping
       +--:(tls) {tls-supported}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
                +---u http-client-grouping
```

Comments:

* The "http-client-stack-grouping" is a convenience grouping for downstream modules. It defines both the "HTTP" and "HTTPS" protocol stacks, with each option enabled by a "feature" statement for application control.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 in this document.

2.1.3. Protocol-accessible Nodes

The "ietf-http-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

2.2. Example Usage

This section presents two examples showing the http-client-grouping populated with some data.

The following example illustrates an HTTP client connecting directly to an HTTP server.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<http-client xmlns="urn:ietf:params:xml:ns:yang:ietf-http-client">
  <client-identity>
    <basic>
      <user-id>bob</user-id>
      <cleartext-password>secret</cleartext-password>
    </basic>
  </client-identity>
</http-client>
```

The following example illustrates the same client connecting through an HTTP proxy. This example is consistent with examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<http-client xmlns="urn:ietf:params:xml:ns:yang:ietf-http-client">
  <client-identity>
    <basic>
      <user-id>bob</user-id>
      <cleartext-password>secret</cleartext-password>
    </basic>
  </client-identity>
  <proxy-connect>
    <https-proxy>
      <tcp-client-parameters>
        <remote-address>corp-fw2.example.com</remote-address>
        <keepalives>
          <idle-time>15</idle-time>
          <max-probes>3</max-probes>
          <probe-interval>30</probe-interval>
        </keepalives>
      </tcp-client-parameters>
      <tls-client-parameters>
        <client-identity>
          <certificate>
            <keystore-reference>
              <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
              <certificate>ex-rsa-cert</certificate>
```

```
              </keystore-reference>
            </certificate>
          </client-identity>
          <server-authentication>
            <ca-certs>
              <truststore-reference>trusted-server-ca-certs</truststor\
   e-reference>
            </ca-certs>
            <ee-certs>
              <truststore-reference>trusted-server-ee-certs</truststor\
   e-reference>
            </ee-certs>
          </server-authentication>
        </tls-client-parameters>
        <http-client-parameters>
          <client-identity>
            <basic>
              <user-id>local-app-1</user-id>
              <cleartext-password>secret</cleartext-password>
            </basic>
          </client-identity>
        </http-client-parameters>
      </https-proxy>
    </proxy-connect>
   </http-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6991].

<CODE BEGINS> file "ietf-http-client@2022-03-07.yang"

```
module ietf-http-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-http-client";
  prefix httpc;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }
```

```
  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
     WG List:  NETCONF WG list <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

  description
    "This module defines reusable groupings for HTTP clients that
     can be used as a basis for specific HTTP client instances.

     Copyright (c) 2021 IETF Trust and the persons identified
     as authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Revised
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC GGGG
     (https://www.rfc-editor.org/info/rfcGGGG); see the RFC
     itself for full legal notices.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
     'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
     'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
     are to be interpreted as described in BCP 14 (RFC 2119)
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.";

  revision 2022-03-07 {
    description
      "Initial version";
```

```
      reference
        "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
    }

    // Features

    feature proxy-connect {
      description
        "Proxy connection configuration is configurable for
         HTTP clients on the server implementing this feature.";
    }

    feature basic-auth {
      description
        "The 'basic-auth' feature indicates that the client
         may be configured to use the 'basic' HTTP authentication
         scheme.";
      reference
        "RFC 7617: The 'Basic' HTTP Authentication Scheme";
    }

    feature tcp-supported {
      description
        "Indicates that the server supports HTTP/TCP.";
    }

    feature tls-supported {
      description
        "Indicates that the server supports HTTP/TLS.";
    }

    // Groupings

    grouping http-client-identity-grouping {
      description
        "A grouping to provide HTTP credentials used by the
         client to authenticate itself to the HTTP server.";
      container client-identity {
        nacm:default-deny-write;
        presence
          "Indicates that a client identity has been configured.
           This statement is present so the mandatory descendant
           nodes do not imply that this node must be configured.";
        description
          "The identity the HTTP client should use when
           authenticating itself to the HTTP server.";
        choice auth-type {
          mandatory true;
```

```
          description
            "A choice amongst available authentication types.";
          case basic {
            container basic {
              if-feature "basic-auth";
              leaf user-id {
                type string;
                mandatory true;
                description
                  "The user-id for the authenticating client.";
              }
              uses ct:password-grouping {
                description
                  "The password for the authenticating client.";
              }
              description
                "The 'basic' HTTP scheme credentials.";
              reference
                "RFC 7617: The 'Basic' HTTP Authentication Scheme";
            }
          }
        }
      }
    } // grouping http-client-identity-grouping

    grouping http-client-grouping {
      description
        "A reusable grouping for configuring a HTTP client.

         This grouping is expected to be used in conjunction with
         other configurations providing, e.g., the hostname or IP
         address and port number the client initiates connections
         to.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'http-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      uses http-client-identity-grouping;

      container proxy-connect {
        nacm:default-deny-write;
        if-feature "proxy-connect";
```

```
        presence
          "Indicates that a proxy server connections have been
           configured.  This statement is present so the mandatory
           descendant nodes do not imply that this node must be
           configured.";
        description
          "Configures the proxy server the HTTP-client is to
           connect thru.";
        choice proxy-type {
          mandatory true;
          description
            "Choice amongst proxy server types.";
          case http {
            container http-proxy {
              description
                "Container for HTTP Proxy (Web Proxy) server
                 configuration parameters.";
              container tcp-client-parameters {
                description
                  "A wrapper around the TCP parameters to avoid
                   name collisions.";
                uses tcpc:tcp-client-grouping;
              }
              container http-client-parameters {
                description
                  "A wrapper around the HTTP parameters to avoid
                   name collisions.";
                uses http-client-identity-grouping;
              }
            }
          }
          case https {
            container https-proxy {
              description
                "Container for HTTPS Proxy (Secure Web Proxy) server
                 configuration parameters.";
              container tcp-client-parameters {
                description
                  "A wrapper around the TCP parameters to avoid
                   name collisions.";
                uses tcpc:tcp-client-grouping;
              }
              container tls-client-parameters {
                description
                  "A wrapper around the TLS parameters to avoid
                   name collisions.";
                uses tlsc:tls-client-grouping;
              }
```

```
              container http-client-parameters {
                description
                  "A wrapper around the HTTP parameters to avoid
                   name collisions.";
                uses http-client-identity-grouping;
              }
            }
          }
        }
      }
    } // grouping http-client-grouping

    grouping http-client-stack-grouping {
      description
        "A grouping that defines common HTTP-based protocol stacks.";
      choice transport {
        mandatory true;
        description
          "Choice amongst various transports type.  TCP, with and
           without TLS are defined here, with 'feature' statements
           so that they may be disabled.  Other transports MAY be
           augmented in as 'case' statements by future efforts.";
        case tcp {
          if-feature "tcp-supported";
          container tcp {
            description
              "Container for TCP-based HTTP protocols.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses tcpc:tcp-client-grouping;
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-client-grouping;
            }
          }
        }
        case tls {
          if-feature "tls-supported";
          container tls {
            description
              "Container for TLS-based HTTP protocols.";
            container tcp-client-parameters {
              description
```

```
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses tcpc:tcp-client-grouping;
            }
            container tls-client-parameters {
              description
                "A wrapper around the TLS parameters to avoid
                 name collisions.";
              uses tlsc:tls-client-grouping;
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-client-grouping;
            }
          }
        }
      }
    } // http-client-stack-grouping

  }
```

<CODE ENDS>

3. The "ietf-http-server" Module

This section defines a YANG 1.1 module called "ietf-http-server". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-http-server" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-http-server" module:

```
Features:
  +-- client-auth-supported
  +-- local-users-supported
  +-- basic-auth
  +-- tcp-supported
  +-- tls-supported
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

3.1.2. Groupings

The "ietf-http-server" module defines the following "grouping" statements:

* http-server-grouping
* http-server-stack-grouping

Each of these groupings are presented in the following subsections.

3.1.2.1. The "http-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-server-grouping" grouping:

```
  grouping http-server-grouping
    +-- server-name?             string
    +-- client-authentication! {client-auth-supported}?
       +-- users {local-users-supported}?
          +-- user* [user-id]
             +-- user-id?       string
             +-- (auth-type)
                +--:(basic)
                   +-- basic {basic-auth}?
                      +-- user-id?    string
                      +-- password?   ianach:crypt-hash
```

Comments:

* The "http-server-grouping" defines the configuration for just "HTTP" part of a protocol stack. It does not, for instance, define any configuration for the "TCP" or "TLS" protocol layers (for that, see Section 3.1.2.2).

* The "server-name" node defines the HTTP server's name, as presented to HTTP clients.

* The "client-authentication" node, which must by enabled by a feature, defines a very simple user-database. Only the "basic" authentication scheme is supported, albeit it must be enabled by a "feature". Other authentication schemes MAY be augmented in.

3.1.2.2. The "http-server-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-server-stack-grouping" grouping:

```
  grouping http-server-stack-grouping
    +-- (transport)
       +--:(tcp) {tcp-supported}?
       |  +-- tcp
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-server-parameters
       |        +---u http-server-grouping
       +--:(tls) {tls-supported}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
                +---u http-server-grouping
```

Comments:

* The "http-server-stack-grouping" is a convenience grouping for downstream modules. It defines both the "HTTP" and "HTTPS" protocol stacks, with each option enabled by a "feature" statement for application control.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-http-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

3.2. Example Usage

This section presents an example showing the http-server-grouping populated with some data.

```
   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <http-server xmlns="urn:ietf:params:xml:ns:yang:ietf-http-server">
     <server-name>foo.example.com</server-name>
   </http-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6991].

```
   <CODE BEGINS> file "ietf-http-server@2022-03-07.yang"

   module ietf-http-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-http-server";
     prefix https;

     import iana-crypt-hash {
       prefix ianach;
       reference
         "RFC 7317: A YANG Data Model for System Management";
     }

     import ietf-netconf-acm {
       prefix nacm;
       reference
         "RFC 8341: Network Configuration Access Control Model";
     }

     import ietf-tcp-server {
       prefix tcps;
       reference
         "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
     }

     import ietf-tls-server {
       prefix tlss;
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";

     contact
       "WG Web:   https://datatracker.ietf.org/wg/netconf
        WG List:  NETCONF WG list <mailto:netconf@ietf.org>
```

```
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

      description
        "This module defines reusable groupings for HTTP servers that
         can be used as a basis for specific HTTP server instances.

         Copyright (c) 2021 IETF Trust and the persons identified
         as authors of the code. All rights reserved.

         Redistribution and use in source and binary forms, with
         or without modification, is permitted pursuant to, and
         subject to the license terms contained in, the Revised
         BSD License set forth in Section 4.c of the IETF Trust's
         Legal Provisions Relating to IETF Documents
         (https://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC GGGG
         (https://www.rfc-editor.org/info/rfcGGGG); see the RFC
         itself for full legal notices.

         The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
         'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
         'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
         are to be interpreted as described in BCP 14 (RFC 2119)
         (RFC 8174) when, and only when, they appear in all
         capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
      }

      // Features

      feature client-auth-supported {
        description
          "Indicates that the configuration for how to authenticate
           clients can be configured herein.  HTTP-level client
           authentication may not be needed when client authentication
           is expected to occur only at another protocol layer.";
      }

      feature local-users-supported {
        description
          "Indicates that the configuration for users can be
           configured herein, as opposed to in an application
```

```
         specific location.";
    }

    feature basic-auth {
      description
        "The 'basic-auth' feature indicates that the server
         may be configured authenticate users using the 'basic'
         HTTP authentication scheme.";
      reference
        "RFC 7617: The 'Basic' HTTP Authentication Scheme";
    }

    feature tcp-supported {
      description
        "Indicates that the server supports HTTP/TCP.";
    }

    feature tls-supported {
      description
        "Indicates that the server supports HTTP/TLS.";
    }

    // Groupings

    grouping http-server-grouping {
      description
        "A reusable grouping for configuring an HTTP server.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'http-server-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      leaf server-name {
        nacm:default-deny-write;
        type string;
        description
          "The value of the 'Server' header field.  If not set, then
           underlying software's default value is used.  Set to the
           empty string to disable.";
      }

      container client-authentication {
        if-feature "client-auth-supported";
```

```
        nacm:default-deny-write;
        presence
          "Indicates that HTTP based client authentication is
           configured.  This statement is present so the mandatory
           descendant nodes do not imply that this node must be
           configured.";
        description
          "Configures how the HTTP server can authenticate HTTP
           clients.  The HTTP server will request that the HTTP
           client send authentication when needed.";
        container users {
          if-feature "local-users-supported";
          description
            "A list of locally configured users.";
          list user {
            key "user-id";
            description
              "The list of local users configured on this device.";
            leaf user-id {
              type string;
              description
                "The user-id for the authenticating client.";
            }
            choice auth-type {
              mandatory true;
              description
                "The authentication type.";
              case basic {
                container basic {
                  if-feature "basic-auth";
                  leaf user-id {
                    type string;
                    description
                      "The user-id for the authenticating client.";
                  }
                  leaf password {
                    nacm:default-deny-write;
                    type ianach:crypt-hash;
                    description
                      "The password for the authenticating client.";
                  }
                  description
                    "The 'basic' HTTP scheme credentials.";
                  reference
                    "RFC 7617:
                      The 'Basic' HTTP Authentication Scheme";
                }
              }
```

```
            }
          }
        }
      } // container client-authentication
    } // grouping http-server-grouping

    grouping http-server-stack-grouping {
      description
        "A grouping that defines common HTTP-based protocol stacks.";
      choice transport {
        mandatory true;
        description
          "Choice amongst various transports type.  TCP, with and
           without TLS are defined here, with 'feature' statements
           so that they may be disabled.  Other transports MAY be
           augmented in as 'case' statements by future efforts.";
        case tcp {
          if-feature "tcp-supported";
          container tcp {
            description
              "Container for TCP-based HTTP protocols.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses tcps:tcp-server-grouping;
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-server-grouping;
            }
          }
        }
        case tls {
          if-feature "tls-supported";
          container tls {
            description
              "Container for TLS-based HTTP protocols.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses tcps:tcp-server-grouping;
            }
            container tls-server-parameters {
              description
```

```
                "A wrapper around the TLS parameters to avoid
                 name collisions.";
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-server-grouping;
            }
          }
        }
      }
    } // http-server-stack-grouping

  }

  <CODE ENDS>
```

4. Security Considerations

4.1. The "ietf-http-client" YANG Module

The "ietf-http-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "client-identity/basic/password" node:

  The cleartext "password" node defined in the "http-client-identity-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

> Please be aware that this module uses groupings from the "ietf-tls-client" and "ietf-tls-server" modules defined in [I-D.ietf-netconf-tls-client-server]. All of the data nodes defined in these groupings have the NACM extension "default-deny-write" set, thus preventing unrestricted write-access to the data nodes defined in those groupings.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

4.2. The "ietf-http-server" YANG Module

The "ietf-http-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

> Please be aware that this module uses groupings from the "ietf-tls-client" and "ietf-tls-server" modules defined in [I-D.ietf-netconf-tls-client-server]. All of the data nodes defined in these groupings have the NACM extension "default-deny-write" set, thus preventing unrestricted write-access to the data nodes defined in those groupings.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-http-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-http-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-http-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-http-client
prefix:       httpc
reference:    RFC GGGG

name:         ietf-http-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-http-server
prefix:       https
reference:    RFC GGGG
```

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Modified Abstract and Intro to be more accurate wrt intended applicability.

* In ietf-http-client, removed "protocol-version" and all auth schemes except "basic".

* In ietf-http-client, factored out "client-identity-grouping" for proxy connections.

* In ietf-http-server, removed "choice required-or-optional" and "choice local-or-external".

* In ietf-http-server, moved the basic auth under a "choice auth-type" limited by new "feature basic-auth".

A.2. 01 to 02

* Removed the unused "external-client-auth-supported" feature from ietf-http-server.

A.3. 02 to 03

* Removed "protocol-versions" from ietf-http-server based on HTTP WG feedback.

* Slightly restructured the "proxy-server" definition in ietf-http-client.

* Added http-client example show proxy server use.

* Added a "Note to Reviewers" note to first page.

A.4. 03 to 04

* Added a parent "container" to "client-identity-grouping" so that it could be better used by the proxy model.

* Added a "choice" to the proxy model enabling selection of proxy types.

* Added 'http-client-stack-grouping' and 'http-server-stack-grouping' convenience groupings.

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.5. 04 to 05

* Fixed titles and a ref in the IANA Considerations section

* Cleaned up examples (e.g., removed FIXMEs)

* Fixed issues found by the SecDir review of the "keystore" draft.

* Updated the "ietf-http-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

A.6. 05 to 06

* Removed note questioning if okay for app to augment-in a 'path' node when needed, discussed during the 108 session.

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.7. 06 to 07

* Added XML-comment above examples explaining the reason for the unusual top-most element's presence.

* Renamed 'client-auth-config-supported' to 'client-auth-supported' consistent with other drafts.

* Wrapped 'container basic' choice inside a 'case basic' per best practice.

* Aligned modules with 'pyang -f' formatting.

* Fixed nits found by YANG Doctor reviews.

A.8. 07 to 08

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

A.9. 08 to 09

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)

* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Ben Schwartz, Mark Nottingham, Rob Wilton (contributor), and Willy Tarreau.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

A YANG Data Model for a Keystore
draft-ietf-netconf-keystore-24

Abstract

This document defines a YANG module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted or hidden. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types

* CCCC --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 8 September 2022.

Table of Contents

1. Introduction 3
1.1. Relation to other RFCs 4
1.2. Specification Language 6
1.3. Terminology 6
1.4. Adherence to the NMDA 6
1.5. Conventions 6
2. The "ietf-keystore" Module 6
2.1. Data Model Overview 7
2.2. Example Usage 14
2.3. YANG Module 26
3. Support for Built-in Keys 35
4. Encrypting Keys in Configuration 37
5. Security Considerations 41
5.1. Security of Data at Rest 41
5.2. Unconstrained Private Key Usage 41
5.3. The "ietf-keystore" YANG Module 41
6. IANA Considerations 42
6.1. The "IETF XML" Registry 42
6.2. The "YANG Module Names" Registry 42
7. References 42

7.1. Normative References 42
7.2. Informative References 43
Appendix A. Change Log 45
A.1. 00 to 01 45
A.2. 01 to 02 45
A.3. 02 to 03 46
A.4. 03 to 04 46
A.5. 04 to 05 46
A.6. 05 to 06 46
A.7. 06 to 07 47
A.8. 07 to 08 47
A.9. 08 to 09 47
A.10. 09 to 10 47
A.11. 10 to 11 48
A.12. 11 to 12 48
A.13. 12 to 13 48
A.14. 13 to 14 48
A.15. 14 to 15 48
A.16. 15 to 16 49
A.17. 16 to 17 49
A.18. 17 to 18 49
A.19. 18 to 19 50
A.20. 19 to 20 50
A.21. 20 to 21 50
A.22. 21 to 22 50
A.23. 22 to 23 50
A.24. 23 to 24 51
Acknowledgements 51
Author's Address 51

1. Introduction

This document defines a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted or hidden (see [I-D.ietf-netconf-crypto-types]. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

The "ietf-keystore" module defines many "grouping" statements intended for use by other modules that may import it. For instance, there are groupings that define enabling a key to be either configured locally (within the defining data model) or be a reference to a key in the keystore.

Special consideration has been given for systems that have cryptographic hardware, such as a Trusted Platform Module (TPM). These systems are unique in that the cryptographic hardware hides the

secret key values. Additionally, such hardware is commonly initialized when manufactured to protect a "built-in" asymmetric key for which the public half is conveyed in an identity certificate (e.g., an IDevID [Std-802.1AR-2018] certificate). Please see Section 3 to see how built-in keys are supported.

This document intends to support existing practices; it does not intend to define new behavior for systems to implement. To simplify implementation, advanced key formats may be selectively implemented.

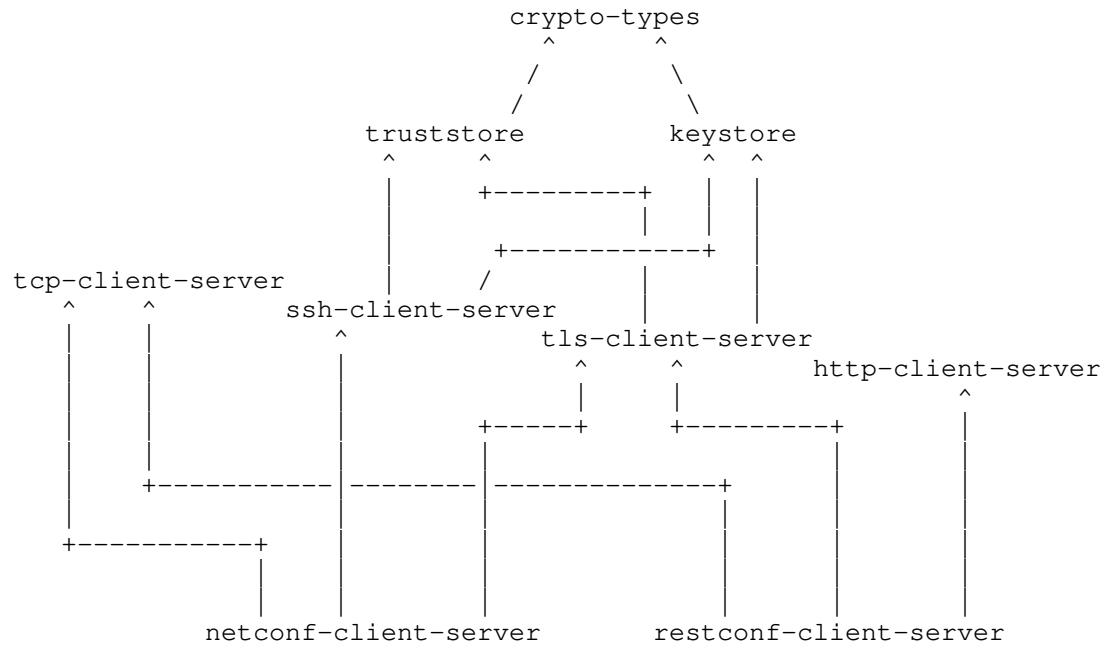
Implementations may utilize zero or more operating system level keystore utilities and/or hardware security modules (HSMs).

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Terminology

The terms "client" and "server" are defined in [RFC6241] and are not redefined here.

The term "keystore" is defined in this draft as a mechanism that intends safeguard secrets placed into it for protection.

The nomenclature "<running>" and "<operational>" are defined in [RFC8342].

The sentence fragments "augmented" and "augmented in" are used herein as the past tense verbified form of the "augment" statement defined in Section 7.17 of [RFC7950].

1.4. Adherence to the NMDA

This document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, keys and associated certificates installed during manufacturing (e.g., for an IDevID certificate) are expected to appear in <operational> (see Section 3).

1.5. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-keystore" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-keystore". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Section 2.2. The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-keystore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-keystore" module:

```
Features:
  +-- central-keystore-supported
  +-- local-definitions-supported
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Typedefs

The following diagram lists the "typedef" statements defined in the "ietf-keystore" module:

```
Typedefs:
  leafref
    +-- symmetric-key-ref
    +-- asymmetric-key-ref
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* All the typedefs defined in the "ietf-keystore" module extend the base "leafref" type defined in [RFC7950].

* The leafrefs refer to symmetric and asymmetric keys in the central keystore, when this module is implemented.

* These typedefs are provided as an aid to downstream modules that import the "ietf-keystore" module.

2.1.3. Groupings

The "ietf-keystore" module defines the following "grouping" statements:

* encrypted-by-choice-grouping

* asymmetric-key-certificate-ref-grouping
* local-or-keystore-symmetric-key-grouping
* local-or-keystore-asymmetric-key-grouping
* local-or-keystore-asymmetric-key-with-certs-grouping
* local-or-keystore-end-entity-cert-with-key-grouping
* keystore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "encrypted-by-choice-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "encrypted-by-choice-grouping" grouping:

```
   |  The grouping's name is intended to be parsed "(encrypted-
   |  by)-(choice)-(grouping)", not as "(encrypted)-(by-
   |  choice)-(grouping)".

  grouping encrypted-by-choice-grouping
    +-- (encrypted-by-choice)
       +--:(symmetric-key-ref)
       |          {central-keystore-supported,symmetric-keys}?
       |  +-- symmetric-key-ref?    ks:symmetric-key-ref
       +--:(asymmetric-key-ref)
                  {central-keystore-supported,asymmetric-keys}?
          +-- asymmetric-key-ref?   ks:asymmetric-key-ref
```

Comments:

* This grouping defines a "choice" statement with options to reference either a symmetric or an asymmetric key configured in the keystore.

* This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations MUST augment in alternate "encrypted-by" references to the alternate locations.

2.1.3.2. The "asymmetric-key-certificate-ref-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "asymmetric-key-certificate-ref-grouping" grouping:

```
  grouping asymmetric-key-certificate-ref-grouping
    +-- asymmetric-key?   ks:asymmetric-key-ref
    |       {central-keystore-supported,asymmetric-keys}?
    +-- certificate?      leafref
```

Comments:

* This grouping defines a reference to a certificate in two parts: the first being the name of the asymmetric key the certificate is associated with, and the second being the name of the certificate itself.

* This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations MAY define an alternate grouping for references to the alternate locations.

2.1.3.3. The "local-or-keystore-symmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-symmetric-key-grouping" grouping:

```
  grouping local-or-keystore-symmetric-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported,symmetric-keys}?
       |  +-- local-definition
       |     +---u ct:symmetric-key-grouping
       +--:(keystore) {central-keystore-supported,symmetric-keys}?
          +-- keystore-reference?   ks:symmetric-key-ref
```

Comments:

* The "local-or-keystore-symmetric-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether a symmetric key is defined locally or as a reference to a symmetric key in the keystore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a symmetric key in an alternate location.

* For the "local-definition" option, the definition uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "symmetric-key-ref" discussed in Section 2.1.2.

2.1.3.4. The "local-or-keystore-asymmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-grouping" grouping:

```
  grouping local-or-keystore-asymmetric-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported,asymmetric-keys}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-grouping
       +--:(keystore) {central-keystore-supported,asymmetric-keys}?
          +-- keystore-reference?   ks:asymmetric-key-ref
```

Comments:

* The "local-or-keystore-asymmetric-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether an asymmetric key is defined locally or as a reference to an asymmetric key in the keystore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference an asymmetric key in an alternate location.

* For the "local-definition" option, the definition uses the "asymmetric-key-pair-grouping" grouping discussed in Section 2.1.4.5 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

2.1.3.5. The "local-or-keystore-asymmetric-key-with-certs-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-with-certs-grouping" grouping:

```
  grouping local-or-keystore-asymmetric-key-with-certs-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported,asymmetric-keys}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-with-certs-grouping
       +--:(keystore) {central-keystore-supported,asymmetric-keys}?
          +-- keystore-reference?   ks:asymmetric-key-ref
```

Comments:

* The "local-or-keystore-asymmetric-key-with-certs-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether an asymmetric key is defined locally or as a reference to an asymmetric key in the keystore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference an asymmetric key in an alternate location.

* For the "local-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

2.1.3.6. The "local-or-keystore-end-entity-cert-with-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-end-entity-cert-with-key-grouping" grouping:

```
  grouping local-or-keystore-end-entity-cert-with-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported,asymmetric-keys}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-with-cert-grouping
       +--:(keystore) {central-keystore-supported,asymmetric-keys}?
          +-- keystore-reference
             +---u asymmetric-key-certificate-ref-grouping
```

Comments:

* The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether a symmetric key is defined locally or as a reference to a symmetric key in the keystore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a symmetric key in an alternate location.

* For the "local-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" uses the "asymmetric-key-certificate-ref-grouping" grouping discussed in Section 2.1.3.2.

2.1.3.7. The "keystore-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "keystore-grouping" grouping:

```
  grouping keystore-grouping
    +-- asymmetric-keys {asymmetric-keys}?
    |  +-- asymmetric-key* [name]
    |     +-- name?                                          string
    |     +---u ct:asymmetric-key-pair-with-certs-grouping
    +-- symmetric-keys {symmetric-keys}?
       +-- symmetric-key* [name]
          +-- name?                          string
          +---u ct:symmetric-key-grouping
```

Comments:

* The "keystore-grouping" grouping defines a keystore instance as being composed of symmetric and asymmetric keys. The structure for the symmetric and asymmetric keys is essentially the same, being a "list" inside a "container".

* For asymmetric keys, each "asymmetric-key" uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].

* For symmetric keys, each "symmetric-key" uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].

2.1.4. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-keystore" module, without expanding the "grouping" statements:

```
module: ietf-keystore
  +--rw keystore
     +---u keystore-grouping
```

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-keystore" module, with all "grouping" statements expanded, enabling the keystore's full structure to be seen:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   module: ietf-keystore
     +--rw keystore
        +--rw asymmetric-keys {asymmetric-keys}?
        |  +--rw asymmetric-key* [name]
        |     +--rw name                                      string
        |     +--rw public-key-format                         identityref
        |     +--rw public-key                                binary
        |     +--rw private-key-format?                       identityref
        |     +--rw (private-key-type)
        |     |  +--:(cleartext-private-key)
        |     |  |  +--rw cleartext-private-key?              binary
        |     |  +--:(hidden-private-key) {hidden-keys}?
        |     |  |  +--rw hidden-private-key?                 empty
        |     |  +--:(encrypted-private-key) {private-key-encryption}?
        |     |     +--rw encrypted-private-key
        |     |        +--rw encrypted-by
        |     |        |  +--rw (encrypted-by-choice)
        |     |        |     +--:(symmetric-key-ref)
        |     |        |     |        {central-keystore-supported,symme\
   tric-keys}?
        |     |        |     |  +--rw symmetric-key-ref?
        |     |        |     |           ks:symmetric-key-ref
        |     |        |     +--:(asymmetric-key-ref)
        |     |        |              {central-keystore-supported,asymm\
   etric-keys}?
        |     |        |        +--rw asymmetric-key-ref?
        |     |        |                 ks:asymmetric-key-ref
        |     |        +--rw encrypted-value-format    identityref
        |     |        +--rw encrypted-value           binary
        |     +--rw certificates
        |     |  +--rw certificate* [name]
        |     |     +--rw name                        string
        |     |     +--rw cert-data                   end-entity-cert-cms
        |     |     +---n certificate-expiration
        |     |              {certificate-expiration-notification}?
        |     |        +-- expiration-date    yang:date-and-time
        |     +---x generate-certificate-signing-request
        |             {certificate-signing-request-generation}?
        |        +---w input
        |        |  +---w csr-info    ct:csr-info
        |        +--ro output
        |           +--ro certificate-signing-request    ct:csr
        +--rw symmetric-keys {symmetric-keys}?
           +--rw symmetric-key* [name]
              +--rw name                    string
              +--rw key-format?             identityref
```

```
              +--rw (key-type)
                 +--:(cleartext-key)
                 |  +--rw cleartext-key?   binary
                 +--:(hidden-key) {hidden-keys}?
                 |  +--rw hidden-key?      empty
                 +--:(encrypted-key) {symmetric-key-encryption}?
                    +--rw encrypted-key
                       +--rw encrypted-by
                       |  +--rw (encrypted-by-choice)
                       |     +--:(symmetric-key-ref)
                       |     |        {central-keystore-supported,symme\
   tric-keys}?
                       |     |  +--rw symmetric-key-ref?
                       |     |           ks:symmetric-key-ref
                       |     +--:(asymmetric-key-ref)
                       |              {central-keystore-supported,asymm\
   etric-keys}?
                       |        +--rw asymmetric-key-ref?
                       |                 ks:asymmetric-key-ref
                       +--rw encrypted-value-format    identityref
                       +--rw encrypted-value           binary
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

* The protocol-accessible nodes for the "ietf-keystore" module are an instance of the "keystore-grouping" grouping discussed in Section 2.1.3.7.

* The reason for why "keystore-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of the keystore to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The examples in this section are encoded using XML, such as might be the case when using the NETCONF protocol. Other encodings MAY be used, such as JSON when using the RESTCONF protocol.

2.2.1. A Keystore Instance

The following example illustrates keys in <running>. Please see Section 3 for an example illustrating built-in values in <operational>.

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <keystore
      xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
      xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

      <symmetric-keys>
         <symmetric-key>
            <name>cleartext-symmetric-key</name>
            <key-format>ct:octet-string-key-format</key-format>
            <cleartext-key>BASE64VALUE=</cleartext-key>
         </symmetric-key>
         <symmetric-key>
            <name>hidden-symmetric-key</name>
            <hidden-key/>
         </symmetric-key>
         <symmetric-key>
            <name>encrypted-symmetric-key</name>
            <key-format>ct:one-symmetric-key-format</key-format>
            <encrypted-key>
              <encrypted-by>
                <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-k\
   ey-ref>
              </encrypted-by>
              <encrypted-value-format>
                ct:cms-enveloped-data-format
              </encrypted-value-format>
              <encrypted-value>BASE64VALUE=</encrypted-value>
            </encrypted-key>
         </symmetric-key>
      </symmetric-keys>

      <asymmetric-keys>
         <asymmetric-key>
            <name>ssh-rsa-key</name>
            <public-key-format>
               ct:ssh-public-key-format
            </public-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>
               ct:rsa-private-key-format
            </private-key-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         </asymmetric-key>
         <asymmetric-key>
            <name>ssh-rsa-key-with-cert</name>
            <public-key-format>
               ct:subject-public-key-info-format
```

```
            </public-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>
               ct:rsa-private-key-format
            </private-key-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
            <certificates>
               <certificate>
                  <name>ex-rsa-cert2</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
         </asymmetric-key>
         <asymmetric-key>
            <name>raw-private-key</name>
            <public-key-format>
               ct:subject-public-key-info-format
            </public-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>
               ct:rsa-private-key-format
            </private-key-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         </asymmetric-key>
         <asymmetric-key>
            <name>rsa-asymmetric-key</name>
            <public-key-format>
               ct:subject-public-key-info-format
            </public-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>
               ct:rsa-private-key-format
            </private-key-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
            <certificates>
               <certificate>
                  <name>ex-rsa-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
         </asymmetric-key>
         <asymmetric-key>
            <name>ec-asymmetric-key</name>
            <public-key-format>
               ct:subject-public-key-info-format
            </public-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>
```

```
              ct:ec-private-key-format
            </private-key-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
            <certificates>
               <certificate>
                  <name>ex-ec-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
         </asymmetric-key>
         <asymmetric-key>
            <name>hidden-asymmetric-key</name>
            <public-key-format>
               ct:subject-public-key-info-format
            </public-key-format>
            <public-key>BASE64VALUE=</public-key>
            <hidden-private-key/>
            <certificates>
               <certificate>
                  <name>builtin-idevid-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
               <certificate>
                  <name>my-ldevid-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
         </asymmetric-key>
         <asymmetric-key>
            <name>encrypted-asymmetric-key</name>
            <public-key-format>
               ct:subject-public-key-info-format
            </public-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>
               ct:one-asymmetric-key-format
            </private-key-format>
            <encrypted-private-key>
              <encrypted-by>
                <symmetric-key-ref>encrypted-symmetric-key</symmetric-k\
   ey-ref>
              </encrypted-by>
              <encrypted-value-format>
                ct:cms-encrypted-data-format
              </encrypted-value-format>
              <encrypted-value>BASE64VALUE=</encrypted-value>
            </encrypted-private-key>
         </asymmetric-key>
```

```
      </asymmetric-keys>
   </keystore>
```

2.2.2. A Certificate Expiration Notification

The following example illustrates a "certificate-expiration" notification for a certificate associated with a key configured in the keystore.

```
   ============== NOTE: '\' line wrapping per RFC 8792 ================

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
       <asymmetric-keys>
         <asymmetric-key>
           <name>hidden-asymmetric-key</name>
           <certificates>
             <certificate>
               <name>my-ldevid-cert</name>
               <certificate-expiration>
                 <expiration-date>2018-08-05T14:18:53-05:00</expiration\
   -date>
               </certificate-expiration>
             </certificate>
           </certificates>
         </asymmetric-key>
       </asymmetric-keys>
     </keystore>
   </notification>
```

2.2.3. The "Local or Keystore" Groupings

This section illustrates the various "local-or-keystore" groupings defined in the "ietf-keystore" module, specifically the "local-or-keystore-symmetric-key-grouping" (Section 2.1.3.3), "local-or-keystore-asymmetric-key-grouping" (Section 2.1.3.4), "local-or-keystore-asymmetric-key-with-certs-grouping" (Section 2.1.3.5), and "local-or-keystore-end-entity-cert-with-key-grouping" (Section 2.1.3.6) groupings.

These examples assume the existence of an example module called "ex-keystore-usage" having the namespace "http://example.com/ns/example-keystore-usage".

The ex-keystore-usage module is first presented using tree diagrams [RFC8340], followed by an instance example illustrating all the "local-or-keystore" groupings in use, followed by the YANG module itself.

The following tree diagram illustrates "ex-keystore-usage" without expanding the "grouping" statements:

```
module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                                                string
     |  +---u ks:local-or-keystore-symmetric-key-grouping
     +--rw asymmetric-key* [name]
     |  +--rw name                                                 string
     |  +---u ks:local-or-keystore-asymmetric-key-grouping
     +--rw asymmetric-key-with-certs* [name]
     |  +--rw name
     |  |       string
     |  +---u ks:local-or-keystore-asymmetric-key-with-certs-grouping
     +--rw end-entity-cert-with-key* [name]
        +--rw name
        |       string
        +---u ks:local-or-keystore-end-entity-cert-with-key-grouping
```

The following tree diagram illustrates the "ex-keystore-usage" module, with all "grouping" statements expanded, enabling the usage's full structure to be seen:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                          string
     |  +--rw (local-or-keystore)
     |     +--:(local) {local-definitions-supported,symmetric-keys}?
     |     |  +--rw local-definition
     |     |     +--rw key-format?            identityref
     |     |     +--rw (key-type)
     |     |        +--:(cleartext-key)
     |     |        |  +--rw cleartext-key?   binary
     |     |        +--:(hidden-key) {hidden-keys}?
     |     |        |  +--rw hidden-key?      empty
     |     |        +--:(encrypted-key) {symmetric-key-encryption}?
     |     |           +--rw encrypted-key
     |     |              +--rw encrypted-by
     |     |              +--rw encrypted-value-format    identityref
```

```
|      |              +--rw encrypted-value           binary
|      +--:(keystore)
|               {central-keystore-supported,symmetric-keys}?
|         +--rw keystore-reference?   ks:symmetric-key-ref
+--rw asymmetric-key* [name]
|  +--rw name                         string
|  +--rw (local-or-keystore)
|     +--:(local) {local-definitions-supported,asymmetric-keys}?
|     |  +--rw local-definition
|     |     +--rw public-key-format              identityref
|     |     +--rw public-key                     binary
|     |     +--rw private-key-format?            identityref
|     |     +--rw (private-key-type)
|     |        +--:(cleartext-private-key)
|     |        |  +--rw cleartext-private-key?   binary
|     |        +--:(hidden-private-key) {hidden-keys}?
|     |        |  +--rw hidden-private-key?      empty
|     |        +--:(encrypted-private-key)
|     |                 {private-key-encryption}?
|     |           +--rw encrypted-private-key
|     |              +--rw encrypted-by
|     |              +--rw encrypted-value-format    identityref
|     |              +--rw encrypted-value           binary
|     +--:(keystore)
|               {central-keystore-supported,asymmetric-keys}?
|         +--rw keystore-reference?   ks:asymmetric-key-ref
+--rw asymmetric-key-with-certs* [name]
|  +--rw name                         string
|  +--rw (local-or-keystore)
|     +--:(local) {local-definitions-supported,asymmetric-keys}?
|     |  +--rw local-definition
|     |     +--rw public-key-format
|     |     |       identityref
|     |     +--rw public-key                                 binary
|     |     +--rw private-key-format?
|     |     |       identityref
|     |     +--rw (private-key-type)
|     |     |  +--:(cleartext-private-key)
|     |     |  |  +--rw cleartext-private-key?            binary
|     |     |  +--:(hidden-private-key) {hidden-keys}?
|     |     |  |  +--rw hidden-private-key?               empty
|     |     |  +--:(encrypted-private-key)
|     |     |           {private-key-encryption}?
|     |     |     +--rw encrypted-private-key
|     |     |        +--rw encrypted-by
|     |     |        +--rw encrypted-value-format    identityref
|     |     |        +--rw encrypted-value           binary
|     |     +--rw certificates
```

```
          |      |      |  +--rw certificate* [name]
          |      |      |     +--rw name                        string
          |      |      |     +--rw cert-data
          |      |      |     |       end-entity-cert-cms
          |      |      |     +---n certificate-expiration
          |      |      |             {certificate-expiration-notification}?
          |      |      |        +-- expiration-date    yang:date-and-time
          |      |      +---x generate-certificate-signing-request
          |      |              {certificate-signing-request-generation}?
          |      |         +---w input
          |      |         |  +---w csr-info    ct:csr-info
          |      |         +--ro output
          |      |            +--ro certificate-signing-request    ct:csr
          |      +--:(keystore)
          |              {central-keystore-supported,asymmetric-keys}?
          |         +--rw keystore-reference?   ks:asymmetric-key-ref
          +--rw end-entity-cert-with-key* [name]
             +--rw name                          string
             +--rw (local-or-keystore)
                +--:(local) {local-definitions-supported,asymmetric-keys}?
                |  +--rw local-definition
                |     +--rw public-key-format
                |     |       identityref
                |     +--rw public-key                                binary
                |     +--rw private-key-format?
                |     |       identityref
                |     +--rw (private-key-type)
                |     |  +--:(cleartext-private-key)
                |     |  |  +--rw cleartext-private-key?            binary
                |     |  +--:(hidden-private-key) {hidden-keys}?
                |     |  |  +--rw hidden-private-key?               empty
                |     |  +--:(encrypted-private-key)
                |     |           {private-key-encryption}?
                |     |     +--rw encrypted-private-key
                |     |        +--rw encrypted-by
                |     |        +--rw encrypted-value-format    identityref
                |     |        +--rw encrypted-value           binary
                |     +--rw cert-data?
                |     |       end-entity-cert-cms
                |     +---n certificate-expiration
                |     |       {certificate-expiration-notification}?
                |     |  +-- expiration-date    yang:date-and-time
                |     +---x generate-certificate-signing-request
                |             {certificate-signing-request-generation}?
                |        +---w input
                |        |  +---w csr-info    ct:csr-info
                |        +--ro output
                |           +--ro certificate-signing-request    ct:csr
                |
```

```
             +--:(keystore)
                      {central-keystore-supported,asymmetric-keys}?
                +--rw keystore-reference
                   +--rw asymmetric-key?   ks:asymmetric-key-ref
                   |       {central-keystore-supported,asymmetric-keys\
}?
                   +--rw certificate?      leafref
```

The following example provides two equivalent instances of each grouping, the first being a reference to a keystore and the second being locally-defined. The instance having a reference to a keystore is consistent with the keystore defined in Section 2.2.1. The two instances are equivalent, as the locally-defined instance example contains the same values defined by the keystore instance referenced by its sibling example.

```
<keystore-usage
  xmlns="http://example.com/ns/example-keystore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- The following two equivalent examples illustrate the -->
  <!-- "local-or-keystore-symmetric-key-grouping" grouping: -->

  <symmetric-key>
    <name>example 1a</name>
    <keystore-reference>cleartext-symmetric-key</keystore-reference>
  </symmetric-key>

  <symmetric-key>
    <name>example 1b</name>
    <local-definition>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-key>BASE64VALUE=</cleartext-key>
    </local-definition>
  </symmetric-key>


  <!-- The following two equivalent examples illustrate the  -->
  <!-- "local-or-keystore-asymmetric-key-grouping" grouping: -->

  <asymmetric-key>
    <name>example 2a</name>
    <keystore-reference>rsa-asymmetric-key</keystore-reference>
  </asymmetric-key>

  <asymmetric-key>
    <name>example 2b</name>
    <local-definition>
```

```
        <public-key-format>
          ct:subject-public-key-info-format
        </public-key-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>
          ct:rsa-private-key-format
        </private-key-format>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      </local-definition>
    </asymmetric-key>


    <!-- the following two equivalent examples illustrate        -->
    <!-- "local-or-keystore-asymmetric-key-with-certs-grouping": -->

    <asymmetric-key-with-certs>
      <name>example 3a</name>
      <keystore-reference>rsa-asymmetric-key</keystore-reference>
    </asymmetric-key-with-certs>

    <asymmetric-key-with-certs>
      <name>example 3b</name>
      <local-definition>
        <public-key-format>
            ct:subject-public-key-info-format
        </public-key-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>
          ct:rsa-private-key-format
        </private-key-format>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
        <certificates>
          <certificate>
            <name>a locally-defined cert</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </certificates>
      </local-definition>
    </asymmetric-key-with-certs>


    <!-- The following two equivalent examples illustrate       -->
    <!-- "local-or-keystore-end-entity-cert-with-key-grouping": -->

    <end-entity-cert-with-key>
      <name>example 4a</name>
      <keystore-reference>
        <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
```

```
        <certificate>ex-rsa-cert</certificate>
      </keystore-reference>
    </end-entity-cert-with-key>

    <end-entity-cert-with-key>
      <name>example 4b</name>
      <local-definition>
        <public-key-format>
          ct:subject-public-key-info-format
        </public-key-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>
          ct:rsa-private-key-format
        </private-key-format>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
        <cert-data>BASE64VALUE=</cert-data>
      </local-definition>
    </end-entity-cert-with-key>

  </keystore-usage>
```

Following is the "ex-keystore-usage" module's YANG definition:

```
  module ex-keystore-usage {
    yang-version 1.1;
    namespace "http://example.com/ns/example-keystore-usage";
    prefix eku;

    import ietf-keystore {
      prefix ks;
      reference
        "RFC CCCC: A YANG Data Model for a Keystore";
    }

    organization
      "Example Corporation";

    contact
      "Author: YANG Designer <mailto:yang.designer@example.com>";

    description
      "This module illustrates notable groupings defined in
       the 'ietf-keystore' module.";

    revision 2022-03-07 {
      description
        "Initial version";
      reference
```

```
        "RFC CCCC: A YANG Data Model for a Keystore";
    }

    container keystore-usage {
      description
        "An illustration of the various keystore groupings.";
      list symmetric-key {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-symmetric-key-grouping;
        description
          "An symmetric key that may be configured locally or be a
           reference to a symmetric key in the keystore.";
      }
      list asymmetric-key {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-grouping;
        description
          "An asymmetric key, with no certs, that may be configured
           locally or be a reference to an asymmetric key in the
           keystore.  The intent is to reference just the asymmetric
           key, not any certificates that may also be associated
           with the asymmetric key.";
      }
      list asymmetric-key-with-certs {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-with-certs-grouping;
        description
          "An asymmetric key and its associated certs, that may be
           configured locally or be a reference to an asymmetric key
           (and its associated certs) in the keystore.";
      }
      list end-entity-cert-with-key {
        key "name";
```

```
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
        description
          "An end-entity certificate and its associated asymmetric
           key, that may be configured locally or be a reference
           to another certificate (and its associated asymmetric
           key) in the keystore.";
      }
    }
  }
```

2.3. YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types].

<CODE BEGINS> file "ietf-keystore@2022-03-07.yang"

```
module ietf-keystore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
  prefix ks;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
     WG List:  NETCONF WG list <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

  description
```

```
       "This module defines a 'keystore' to centralize management
        of security credentials.

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC CCCC
        (https://www.rfc-editor.org/info/rfcCCCC); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

     revision 2022-03-07 {
       description
         "Initial version";
       reference
         "RFC CCCC: A YANG Data Model for a Keystore";
     }

     /****************/
     /*   Features   */
     /****************/

     feature central-keystore-supported {
       description
         "The 'central-keystore-supported' feature indicates that
          the server supports the keystore (i.e., implements the
          'ietf-keystore' module).";
     }

     feature local-definitions-supported {
       description
         "The 'local-definitions-supported' feature indicates that
          the server supports locally-defined keys.";
     }
```

```
  feature asymmetric-keys {
    description
      "The 'asymmetric-keys' feature indicates that the server
       supports asymmetric keys in keystores.";
  }

  feature symmetric-keys {
    description
      "The 'symmetric-keys' feature indicates that the server
       supports symmetric keys in keystores.";
  }

  /****************/
  /*   Typedefs   */
  /****************/

  typedef symmetric-key-ref {
    type leafref {
      path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key"
         + "/ks:name";
    }
    description
      "This typedef enables modules to easily define a reference
       to a symmetric key stored in the keystore, when this
       module is implemented.";
  }

  typedef asymmetric-key-ref {
    type leafref {
      path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
         + "/ks:name";
    }
    description
      "This typedef enables modules to easily define a reference
       to an asymmetric key stored in the keystore, when this
       module is implemented.";
  }

  /*****************/
  /*   Groupings   */
  /*****************/

  grouping encrypted-by-choice-grouping {
    description
      "A grouping that defines a 'choice' statement that can be
       augmented into the 'encrypted-by' node, present in the
       'symmetric-key-grouping' and 'asymmetric-key-pair-grouping'
       groupings defined in RFC AAAA, enabling references to keys
```

```
          in the keystore, when this module is implemented.";
        choice encrypted-by-choice {
          nacm:default-deny-write;
          mandatory true;
          description
            "A choice amongst other symmetric or asymmetric keys.";
          case symmetric-key-ref {
            if-feature "central-keystore-supported";
            if-feature "symmetric-keys";
            leaf symmetric-key-ref {
              type ks:symmetric-key-ref;
              description
                "Identifies the symmetric key used to encrypt the
                 associated key.";
            }
          }
          case asymmetric-key-ref {
            if-feature "central-keystore-supported";
            if-feature "asymmetric-keys";
            leaf asymmetric-key-ref {
              type ks:asymmetric-key-ref;
              description
                "Identifies the asymmetric key whose public key
                 encrypted the associated key.";
            }
          }
        }
      }

      grouping asymmetric-key-certificate-ref-grouping {
        description
          "This grouping defines a reference to a specific certificate
           associated with an asymmetric key stored in the keystore,
           when this module is implemented.";
        leaf asymmetric-key {
          nacm:default-deny-write;
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          type ks:asymmetric-key-ref;
          must '../certificate';
          description
            "A reference to an asymmetric key in the keystore.";
        }
        leaf certificate {
          nacm:default-deny-write;
          type leafref {
            path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
               + "[ks:name = current()/../asymmetric-key]/"
```

```
            + "ks:certificates/ks:certificate/ks:name";
        }
        must '../asymmetric-key';
        description
          "A reference to a specific certificate of the
           asymmetric key in the keystore.";
      }
    }

    // local-or-keystore-* groupings

    grouping local-or-keystore-symmetric-key-grouping {
      description
        "A grouping that expands to allow the symmetric key to be
         either stored locally, i.e., within the using data model,
         or a reference to a symmetric key stored in the keystore.

         Servers that do not 'implement' this module, and hence
         'central-keystore-supported' is not defined, SHOULD
         augment in custom 'case' statements enabling references
         to the alternate keystore locations.";
      choice local-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case local {
          if-feature "local-definitions-supported";
          if-feature "symmetric-keys";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses ct:symmetric-key-grouping;
          }
        }
        case keystore {
          if-feature "central-keystore-supported";
          if-feature "symmetric-keys";
          leaf keystore-reference {
            type ks:symmetric-key-ref;
            description
              "A reference to an symmetric key that exists in
               the keystore, when this module is implemented.";
          }
        }
      }
    }
```

```
    grouping local-or-keystore-asymmetric-key-grouping {
      description
        "A grouping that expands to allow the asymmetric key to be
         either stored locally, i.e., within the using data model,
         or a reference to an asymmetric key stored in the keystore.

         Servers that do not 'implement' this module, and hence
         'central-keystore-supported' is not defined, SHOULD
         augment in custom 'case' statements enabling references
         to the alternate keystore locations.";
      choice local-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case local {
          if-feature "local-definitions-supported";
          if-feature "asymmetric-keys";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses ct:asymmetric-key-pair-grouping;
          }
        }
        case keystore {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          leaf keystore-reference {
            type ks:asymmetric-key-ref;
            description
              "A reference to an asymmetric key that exists in
               the keystore, when this module is implemented.  The
               intent is to reference just the asymmetric key
               without any regard for any certificates that may
               be associated with it.";
          }
        }
      }
    }

    grouping local-or-keystore-asymmetric-key-with-certs-grouping {
      description
        "A grouping that expands to allow an asymmetric key and
         its associated certificates to be either stored locally,
         i.e., within the using data model, or a reference to an
         asymmetric key (and its associated certificates) stored
         in the keystore.
```

```
          Servers that do not 'implement' this module, and hence
          'central-keystore-supported' is not defined, SHOULD
          augment in custom 'case' statements enabling references
          to the alternate keystore locations.";
        choice local-or-keystore {
          nacm:default-deny-write;
          mandatory true;
          description
            "A choice between an inlined definition and a definition
             that exists in the keystore.";
          case local {
            if-feature "local-definitions-supported";
            if-feature "asymmetric-keys";
            container local-definition {
              description
                "Container to hold the local key definition.";
              uses ct:asymmetric-key-pair-with-certs-grouping;
            }
          }
          case keystore {
            if-feature "central-keystore-supported";
            if-feature "asymmetric-keys";
            leaf keystore-reference {
              type ks:asymmetric-key-ref;
              description
                "A reference to an asymmetric-key (and all of its
                 associated certificates) in the keystore, when
                 this module is implemented.";
            }
          }
        }
      }

      grouping local-or-keystore-end-entity-cert-with-key-grouping {
        description
          "A grouping that expands to allow an end-entity certificate
           (and its associated asymmetric key pair) to be either stored
           locally, i.e., within the using data model, or a reference
           to a specific certificate in the keystore.

           Servers that do not 'implement' this module, and hence
           'central-keystore-supported' is not defined, SHOULD
           augment in custom 'case' statements enabling references
           to the alternate keystore locations.";
        choice local-or-keystore {
          nacm:default-deny-write;
          mandatory true;
          description
```

```
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case local {
          if-feature "local-definitions-supported";
          if-feature "asymmetric-keys";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses ct:asymmetric-key-pair-with-cert-grouping;
          }
        }
        case keystore {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          container keystore-reference {
            uses asymmetric-key-certificate-ref-grouping;
            description
              "A reference to a specific certificate associated with
               an asymmetric key stored in the keystore, when this
               module is implemented.";
          }
        }
      }
    }

    grouping keystore-grouping {
      description
        "Grouping definition enables use in other contexts.  If ever
         done, implementations MUST augment new 'case' statements
         into the various local-or-keystore 'choice' statements to
         supply leafrefs to the model-specific location(s).";
      container asymmetric-keys {
        nacm:default-deny-write;
        if-feature "asymmetric-keys";
        description
          "A list of asymmetric keys.";
        list asymmetric-key {
          key "name";
          description
            "An asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the asymmetric key.";
          }
          uses ct:asymmetric-key-pair-with-certs-grouping;
        }
      }
```

```
      container symmetric-keys {
        nacm:default-deny-write;
        if-feature "symmetric-keys";
        description
          "A list of symmetric keys.";
        list symmetric-key {
          key "name";
          description
            "A symmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the symmetric key.";
          }
          uses ct:symmetric-key-grouping;
        }
      }
    }

    /*********************************/
    /*   Protocol accessible nodes   */
    /*********************************/

    container keystore {
      description
        "A central keystore containing a list of symmetric keys and
         a list of asymmetric keys.";
      nacm:default-deny-write;
      uses keystore-grouping {
        augment "symmetric-keys/symmetric-key/key-type/encrypted-key/"
              + "encrypted-key/encrypted-by" {
          description
            "Augments in a choice statement enabling the encrypting
             key to be any other symmetric or asymmetric key in the
             central keystore.";
          uses encrypted-by-choice-grouping;
        }
        augment "asymmetric-keys/asymmetric-key/private-key-type/"
              + "encrypted-private-key/encrypted-private-key/"
              + "encrypted-by" {
          description
            "Augments in a choice statement enabling the encrypting
             key to be any other symmetric or asymmetric key in the
             central keystore.";
          uses encrypted-by-choice-grouping;
        }
      }
    }
```

```
}

<CODE ENDS>
```

3. Support for Built-in Keys

In some implementations, a server may support built-in keys. Built-in keys MAY be set during the manufacturing process or be dynamically generated the first time the server is booted or a particular service (e.g., SSH) is enabled.

The primary characteristic of the built-in keys is that they are provided by the system, as opposed to configuration. As such, they are present in <operational> (and <system> [I-D.ma-netmod-with-system], if used). The example below illustrates what the keystore in <operational> might look like for a server in its factory default state.

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

In order for the built-in keys (and their associated built-in certificates) to be referenced by configuration, the referenced keys and associated certificates MUST first be copied into <running>.

Built-in keys that are "hidden" MUST be copied into <running> using the same key values, so that the server can bind them to the built-in entries.

Built-in keys that are "encrypted" MAY be copied into other parts of the configuration so long as they are otherwise unmodified (e.g., the "encrypted-by" reference cannot be altered).

Built-in keys that are "cleartext" MAY be copied into other parts of the configuration but, by doing so, they lose their association to the built-in entries and any assurances afforded by knowing they are/were built-in.

The built-in keys and built-in associated certificates are immutable by configuration operations. With exception to additional/custom certificates associated to a built-in key, servers MUST ignore attempts to modify any aspect of built-in keys and/or built-in associated certificates.

The following example illustrates how a single built-in key definition from the previous example has been propagated to <running>:

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <asymmetric-keys>
    <asymmetric-key>
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

After the above configuration is applied, <operational> should appear as follows:

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate or:origin="or:intended">
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

4. Encrypting Keys in Configuration

This section describes an approach that enables both the symmetric and asymmetric keys on a server to be encrypted, such that traditional backup/restore procedures can be used without concern for the keys being compromised when in transit.

4.1. Key Encryption Key

The ability to encrypt configured keys is predicated on the existence of a "key encryption key" (KEK). There may be any number of KEKs in a system. A KEK, by its namesake, is a key that is used to encrypt other keys. A KEK MAY be either a symmetric key or an asymmetric key.

If a KEK is a symmetric key, then the server MUST provide an API for administrators to encrypt other keys without needing to know the symmetric key's value. If the KEK is an asymmetric key, then the server MAY provide an API enabling the encryption of other keys or, alternatively, let the administrators do so themselves using the asymmetric key's public half.

A server MUST possess (or be able to possess, in case the KEK has been encrypted by another KEK) a KEK's cleartext value so that it can decrypt the other keys in the configuration at runtime.

4.2. Configuring Encrypted Keys

Each time a new key is configured, it SHOULD be encrypted by a KEK.

In "ietf-crypto-types" [I-D.ietf-netconf-crypto-types], the format for encrypted values is described by identity statements derived from the "symmetrically-encrypted-value-format" and "symmetrically-encrypted-value-format" identity statements.

Implementations SHOULD provide an API that simultaneously generates and encrypts a key (symmetric or asymmetric) using a KEK. Thusly newly generated key cleartext values may never known to the administrators generating the keys.

In case the server implementation does not provide such an API, then the generating and encrypting steps MAY be performed outside the server, e.g., by an administrator with special access control rights (e.g., an organization's crypto officer).

In either case, the encrypted key can be configured into the keystore using either the "encrypted-key" (for symmetric keys) or the "encrypted-private-key" (for asymmetric keys) nodes. These two nodes contain both the encrypted value as well as a reference to the KEK that encrypted the key.

4.3. Migrating Configuration to Another Server

When a KEK is used to encrypt other keys, migrating the configuration to another server is only possible if the second server has the same KEK. How the second server comes to have the same KEK is discussed in this section.

In some deployments, mechanisms outside the scope of this document may be used to migrate a KEK from one server to another. That said, beware that the ability to do so typically entails having access to the first server but, in many scenarios, the first server may no longer be operational.

In other deployments, an organization's crypto officer, possessing a KEK's cleartext value, configures the same KEK on the second server, presumably as a hidden key or a key protected by access-control (e.g., NACM's "default-deny-all"), so that the cleartext value is not disclosed to regular administrators. However, this approach creates high-coupling to and dependency on the crypto officers that does not scale in production environments.

In order to decouple the crypto officers from the regular administrators, a special KEK, called the "master key" (MK), may be used.

A MK is commonly a globally-unique built-in (see Section 3) asymmetric key. The private key, due to its long lifetime, is hidden (i.e., "hidden-private-key" in Section 2.1.4.5. of [I-D.ietf-netconf-crypto-types]). The public key is often contained in an identity certificate (e.g., IDevID). How to configure a MK during the manufacturing process is outside the scope of this document.

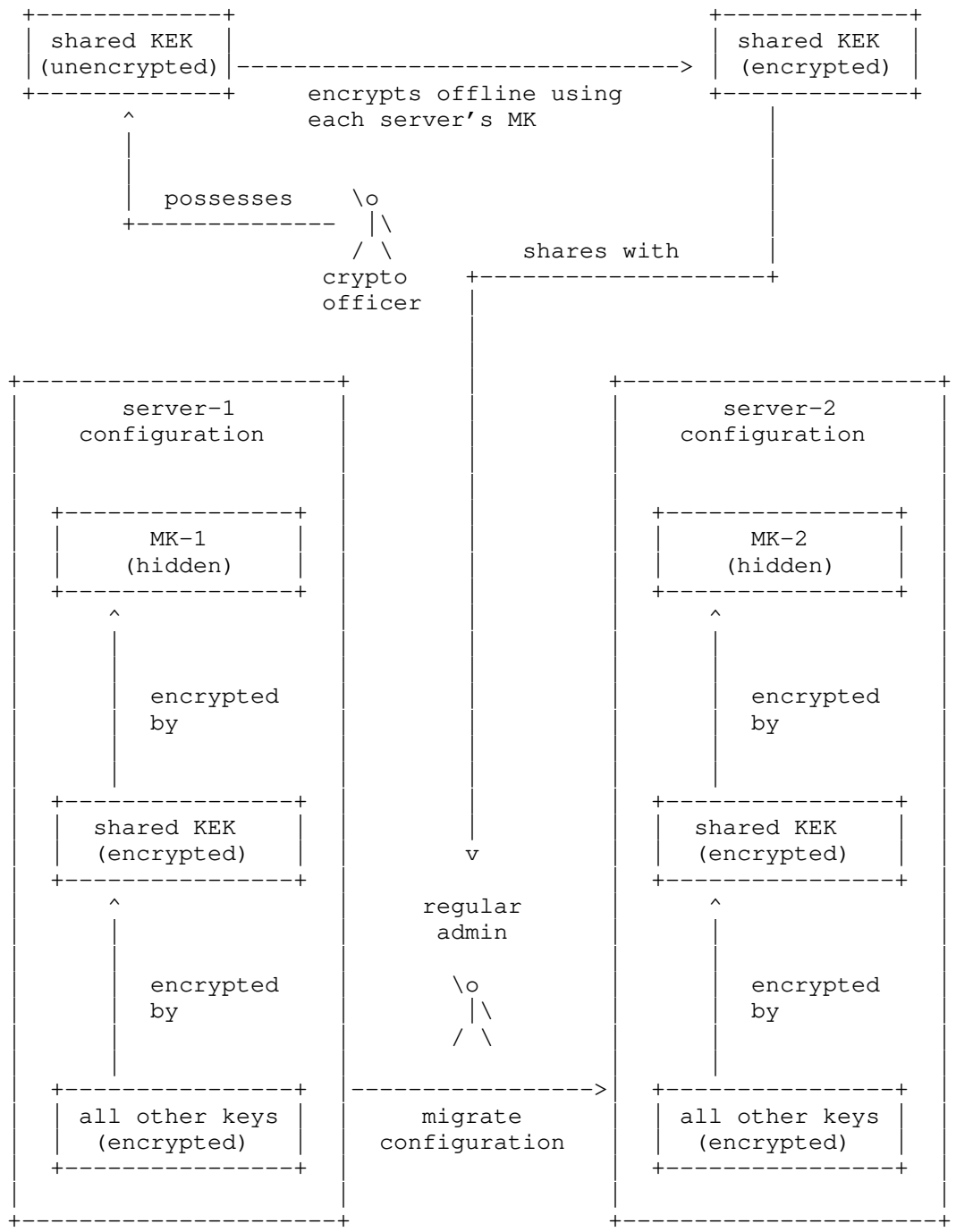
It is RECOMMENDED that MKs are built-in and hidden but, if this is not possible, access control mechanisms like NACM SHOULD be used to limit access to the MK's secret data only to the most trusted authorized clients (e.g., an organization's crypto officer). In this case, it is RECOMMENDED that the MK is not built-in and hence is, effectively, just like a KEK.

Assuming the server has a MK, the MK can be used to encrypt a "shared KEK", which is then used to encrypt the keys configured by regular administrators.

With this extra level of indirection, it is possible for a crypto officer to encrypt the same KEK for a multiplicity of servers offline using the public key contained in their identity certificates. The crypto officer can then safely handoff the encrypted KEKs to the regular administrators responsible for server installations, including migrations.

In order to migrate the configuration from a first server, an administrator would need to make just a single modification to the configuration before loading it onto a second server, which is to replace the encrypted KEK keystore entry from the first server with the encrypted KEK for the second server. Upon doing this, the configuration (containing many encrypted keys) can be loaded into the second server while enabling the second server to decrypt all the encrypted keys in the configuration.

The following diagram illustrates this idea:


shared KEK
(unencrypted)
encrypts offline using
each server's MK
shared KEK
(encrypted)
possesses
crypto
officer
shares with
server-1
configuration
MK-1
(hidden)
encrypted
by
shared KEK
(encrypted)
all other keys
(encrypted)
server-2
configuration
MK-2
(hidden)
encrypted
by
shared KEK
(encrypted)
all other keys
(encrypted)
regular
admin
migrate
configuration

5. Security Considerations

5.1. Security of Data at Rest

The YANG module defined in this document defines a mechanism called a "keystore" that, by its name, suggests that it will protect its contents from unauthorized disclosure and modification.

Security controls for the API (i.e., data in motion) are discussed in Section 5.3, but controls for the data at rest cannot be specified by the YANG module.

In order to satisfy the expectations of a "keystore", it is RECOMMENDED that implementations ensure that the keystore contents are encrypted when persisted to non-volatile memory.

5.2. Unconstrained Private Key Usage

This module enables the configuration of private keys without constraints on their usage, e.g., what operations the key is allowed to be used for (e.g., signature, decryption, both).

This module also does not constrain the usage of the associated public keys, other than in the context of a configured certificate (e.g., an identity certificate), in which case the key usage is constrained by the certificate.

5.3. The "ietf-keystore" YANG Module

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

> Please be aware that this module uses the "cleartext-key" and "cleartext-private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing uncontrolled read-access to the cleartext key values.

All the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "keystore" instance, may be considered sensitive or vulnerable in some network environments.. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any "rpc" or "action" statements, and thus the security considerations for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-keystore
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

6.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registration is requested:

```
name:         ietf-keystore
namespace:    urn:ietf:params:xml:ns:yang:ietf-keystore
prefix:       ks
reference:    RFC CCCC
```

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[I-D.ma-netmod-with-system]
Ma, Q., Watsen, K., Wu, Q., Chong, F., and J. Lindblad, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ma-netmod-with-system-02, 14 February 2022, <https://datatracker.ietf.org/doc/html/draft-ma-netmod-with-system-02>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[Std-802.1AR-2018] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", August 2018, <https://standards.ieee.org/standard/802_1AR-2018.html>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Replaced the 'certificate-chain' structures with PKCS#7 structures. (Issue #1)

* Added 'private-key' as a configurable data node, and removed the 'generate-private-key' and 'load-private-key' actions. (Issue #2)

* Moved 'user-auth-credentials' to the ietf-ssh-client module. (Issues #4 and #5)

A.2. 01 to 02

* Added back 'generate-private-key' action.

* Removed 'RESTRICTED' enum from the 'private-key' leaf type.

* Fixed up a few description statements.

A.3. 02 to 03

* Changed draft's title.
* Added missing references.
* Collapsed sections and levels.
* Added RFC 8174 to Requirements Language Section.
* Renamed 'trusted-certificates' to 'pinned-certificates'.
* Changed 'public-key' from config false to config true.
* Switched 'host-key' from OneAsymmetricKey to definition from RFC 4253.

A.4. 03 to 04

* Added typedefs around leafrefs to common keystore paths
* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Removed Design Considerations section
* Moved key and certificate definitions from data tree to groupings

A.5. 04 to 05

* Removed trust anchors (now in their own draft)
* Added back global keystore structure
* Added groupings enabling keys to either be locally defined or a reference to the keystore.

A.6. 05 to 06

* Added feature "local-keys-supported"
* Added nacm:default-deny-all and nacm:default-deny-write
* Renamed generate-asymmetric-key to generate-hidden-key
* Added an install-hidden-key action

* Moved actions inside fo the "asymmetric-key" container

* Moved some groupings to draft-ietf-netconf-crypto-types

A.7. 06 to 07

* Removed a "require-instance false"

* Clarified some description statements

* Improved the keystore-usage examples

A.8. 07 to 08

* Added "local-definition" containers to avoid posibility of the action/notification statements being under a "case" statement.

* Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

A.9. 08 to 09

* Added a 'description' statement to the 'must' in the /keystore/asymmetric-key node explaining that the descendant values may exist in <operational> only, and that implementation MUST assert that the values are either configured or that they exist in <operational>.

* Copied above 'must' statement (and description) into the local-or-keystore-asymmetric-key-grouping, local-or-keystore-asymmetric-key-with-certs-grouping, and local-or-keystore-end-entity-cert-with-key-grouping statements.

A.10. 09 to 10

* Updated draft title to match new truststore draft title

* Moved everything under a top-level 'grouping' to enable use in other contexts.

* Renamed feature from 'local-keys-supported' to 'local-definitions-supported' (same name used in truststore)

* Removed the either-all-or-none 'must' expressions for the key's 3-tuple values (since the values are now 'mandatory true' in crypto-types)

* Example updated to reflect 'mandatory true' change in crypto-types draft

A.11. 10 to 11

* Replaced typedef asymmetric-key-certificate-ref with grouping asymmetric-key-certificate-ref-grouping.

* Added feature feature 'key-generation'.

* Cloned groupings symmetric-key-grouping, asymmetric-key-pair-grouping, asymmetric-key-pair-with-cert-grouping, and asymmetric-key-pair-with-certs-grouping from crypto-keys, augmenting into each new case statements for values that have been encrypted by other keys in the keystore. Refactored keystore model to use these groupings.

* Added new 'symmetric-keys' lists, as a sibling to the existing 'asymmetric-keys' list.

* Added RPCs (not actions) 'generate-symmetric-key' and 'generate-asymmetric-key' to *return* a (potentially encrypted) key.

A.12. 11 to 12

* Updated to reflect crypto-type's draft using enumerations over identities.

* Added examples for the 'generate-symmetric-key' and 'generate-asymmetric-key' RPCs.

* Updated the Introduction section.

A.13. 12 to 13

* Updated examples to incorporate new "key-format" identities.

* Made the two "generate-*-key" RPCs be "action" statements instead.

A.14. 13 to 14

* Updated YANG module and examples to incorporate the new iana-*-algorithm modules in the crypto-types draft..

A.15. 14 to 15

* Added new "Support for Built-in Keys" section.

* Added 'must' expressions asserting that the 'key-format' leaf whenever an encrypted key is specified.

* Added local-or-keystore-symmetric-key-grouping for PSK support.

A.16. 15 to 16

* Moved the generate key actions to ietf-crypt-types as RPCs, which are augmented by ietf-keystore to support encrypted keys. Examples updated accordingly.

* Added a SSH certificate-based key (RFC 6187) and a raw private key to the example instance document (partly so they could be referenced by examples in the SSH and TLS client/server drafts.

A.17. 16 to 17

* Removed augments to the "generate-symmetric-key" and "generate-asymmetric-key" groupings.

* Removed "generate-symmetric-key" and "generate-asymmetric-key" examples.

* Removed the "algorithm" nodes from remaining examples.

* Updated the "Support for Built-in Keys" section.

* Added new section "Encrypting Keys in Configuration".

* Added a "Note to Reviewers" note to first page.

A.18. 17 to 18

* Removed dangling/unnecessary ref to RFC 8342.

* r/MUST/SHOULD/ wrt strength of keys being configured over transports.

* Added an example for the "certificate-expiration" notification.

* Clarified that OS MAY have a multiplicity of underlying keystores and/or HSMs.

* Clarified expected behavior for "built-in" keys in <operational>

* Clarified the "Migrating Configuration to Another Server" section.

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.19. 18 to 19

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

A.20. 19 to 20

* Addressed SecDir comments from Magnus Nystroem and Sandra Murphy.

A.21. 20 to 21

* Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir.

* (Editorial) Removed the output of "grouping" statements in the tree diagrams for the "ietf-keystore" and "ex-keystore-usage" modules.

* Addressed comments raised by YANG Doctor.

A.22. 21 to 22

* Added prefixes to 'path' statements per trust-anchors/issues/1

* Renamed feature "keystore-supported" to "central-keystore-supported".

* Associated with above, generally moved text to refer to a "central" keystore.

* Aligned modules with `pyang -f` formatting.

* Fixed nits found by YANG Doctor reviews.

A.23. 22 to 23

* Updated 802.1AR ref to latest version

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

A.24. 23 to 24

* Added features "asymmetric-keys" and "symmetric-keys"
* fixup the 'WG Web' and 'WG List' lines in YANG module(s)
* fixup copyright (i.e., s/Simplified/Revised/) in YANG module(s)
* Added Informative reference to ma-netmod-with-system

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Benoit Claise, Bert Wijnen, Balazs Kovacs, David Lamparter, Eric Voit, Ladislav Lhotka, Liang Xia, Juergen Schoenwaelder, Mahesh Jethanandani, Magnus Nystroem, Martin Bjoerklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Reshad Rahman, Sandra Murphy, Sean Turner, and Tom Petch.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-25

Abstract

This document defines two YANG modules, one module to configure a NETCONF client and the other module to configure a NETCONF server. Both modules support both the SSH and TLS transport protocols, and support both standard NETCONF and NETCONF Call Home connections.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements (note: not all may be present):

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* EEEE --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
* GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server
* HHHH --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Table of Contents

1. Introduction 4
  1.1. Relation to other RFCs 4
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
  1.4. Conventions 5
2. The "ietf-netconf-client" Module 6

2.1. Data Model Overview 6
2.2. Example Usage 10
2.3. YANG Module 14
3. The "ietf-netconf-server" Module 25
3.1. Data Model Overview 25
3.2. Example Usage 30
3.3. YANG Module 36
4. Security Considerations 50
4.1. The "ietf-netconf-client" YANG Module 50
4.2. The "ietf-netconf-server" YANG Module 51
5. IANA Considerations 51
5.1. The "IETF XML" Registry 51
5.2. The "YANG Module Names" Registry 52
6. References 52
6.1. Normative References 52
6.2. Informative References 53
Appendix A. Change Log 55
A.1. 00 to 01 55
A.2. 01 to 02 55
A.3. 02 to 03 55
A.4. 03 to 04 55
A.5. 04 to 05 56
A.6. 05 to 06 56
A.7. 06 to 07 56
A.8. 07 to 08 56
A.9. 08 to 09 56
A.10. 09 to 10 57
A.11. 10 to 11 57
A.12. 11 to 12 57
A.13. 12 to 13 57
A.14. 13 to 14 58
A.15. 14 to 15 58
A.16. 15 to 16 58
A.17. 16 to 17 58
A.18. 17 to 18 58
A.19. 18 to 19 58
A.20. 19 to 20 59
A.21. 20 to 21 59
A.22. 21 to 22 59
A.23. 22 to 23 59
A.24. 23 to 24 59
A.25. 24 to 25 59
Acknowledgements 60
Author's Address 60

1. Introduction

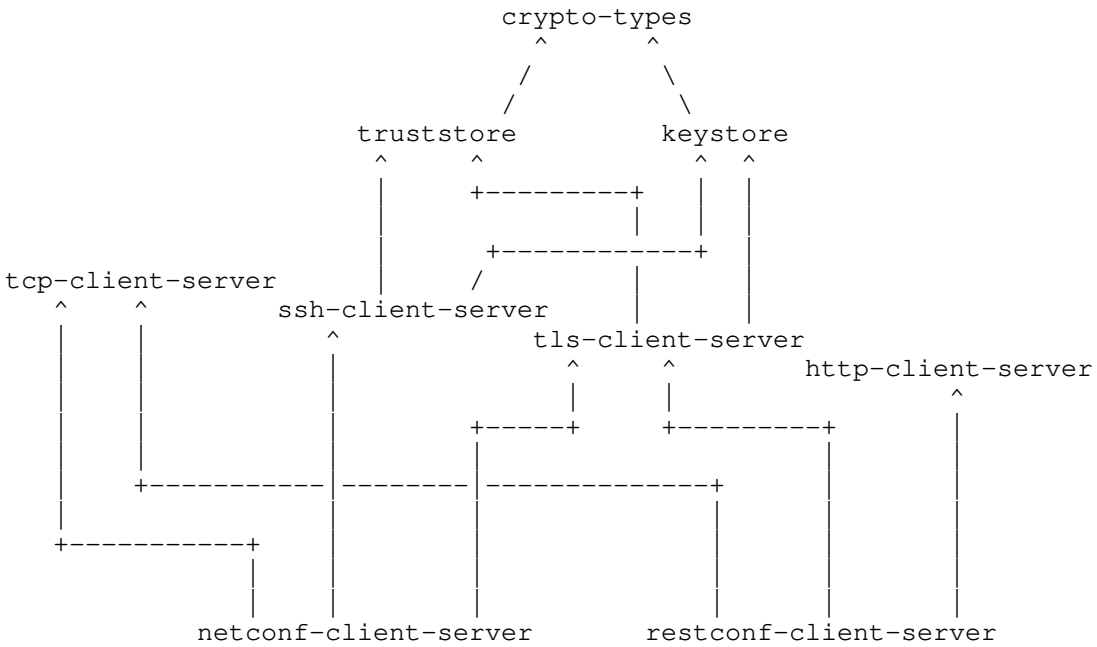
This document defines two YANG [RFC7950] modules, one module to configure a NETCONF [RFC6241] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [RFC6242] and NETCONF over TLS [RFC7589] and NETCONF Call Home connections [RFC8071].

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-netconf-client" Module

The NETCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the NETCONF client supports.

2.1. Data Model Overview

This section provides an overview of the "ietf-netconf-client" module in terms of its features and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-netconf-client" module:

```
Features:
  +-- ssh-initiate
  +-- tls-initiate
  +-- ssh-listen
  +-- tls-listen
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Groupings

The "ietf-netconf-client" module defines the following "grouping" statements:

* netconf-client-grouping
* netconf-client-initiate-stack-grouping
* netconf-client-listen-stack-grouping
* netconf-client-app-grouping

Each of these groupings are presented in the following subsections.

2.1.2.1. The "netconf-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-grouping" grouping:

```
  grouping netconf-client-grouping ---> <empty>
```

Comments:

* This grouping does not define any nodes, but is maintained so that downstream modules can augment nodes into it if needed.

* The "netconf-client-grouping" defines, if it can be called that, the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

2.1.2.2. The "netconf-client-initiate-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-initiate-stack-grouping" grouping:

```
  grouping netconf-client-initiate-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-initiate}?
       |  +-- ssh
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- ssh-client-parameters
       |     |  +---u sshc:ssh-client-grouping
       |     +-- netconf-client-parameters
       |        +--u ncc:netconf-client-grouping
       +--:(tls) {tls-initiate}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- netconf-client-parameters
                +---u ncc:netconf-client-grouping
```

Comments:

* The "netconf-client-initiate-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that initiate connections to NETCONF servers, as opposed to receiving call-home [RFC8071] connections.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

- The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
- The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
- The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
- The "netconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.3. The "netconf-client-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-listen-stack-grouping" grouping:

```
  grouping netconf-client-listen-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-listen}?
       |  +-- ssh
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- ssh-client-parameters
       |     |  +---u sshc:ssh-client-grouping
       |     +-- netconf-client-parameters
       |        +--u ncc:netconf-client-grouping
       +--:(tls) {tls-listen}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- netconf-client-parameters
                +---u ncc:netconf-client-grouping
```

Comments:

* The "netconf-client-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that receive call-home [RFC8071] connections from NETCONF servers.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].

- The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
- The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
- The "netconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.4. The "netconf-client-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-app-grouping" grouping:

```
  grouping netconf-client-app-grouping
    +-- initiate! {ssh-initiate or tls-initiate}?
    |  +-- netconf-server* [name]
    |     +-- name?                 string
    |     +-- endpoints
    |     |  +-- endpoint* [name]
    |     |     +-- name?                                        string
    |     |     +---u netconf-client-initiate-stack-grouping
    |     +-- connection-type
    |     |  +-- (connection-type)
    |     |     +--:(persistent-connection)
    |     |     |  +-- persistent!
    |     |     +--:(periodic-connection)
    |     |        +-- periodic!
    |     |           +-- period?         uint16
    |     |           +-- anchor-time?    yang:date-and-time
    |     |           +-- idle-timeout?   uint16
    |     +-- reconnect-strategy
    |        +-- start-with?     enumeration
    |        +-- max-attempts?   uint8
    +-- listen! {ssh-listen or tls-listen}?
       +-- idle-timeout?   uint16
       +-- endpoint* [name]
          +-- name?                                    string
          +---u netconf-client-listen-stack-grouping
```

Comments:

* The "netconf-client-app-grouping" defines the configuration for a NETCONF client that supports both initiating connections to NETCONF servers as well as receiving call-home connections from NETCONF servers.

* Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "netconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.
  - The "netconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

2.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-netconf-client" module:

```
module: ietf-netconf-client
  +--rw netconf-client
     +---u netconf-client-app-grouping
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

* For the "ietf-netconf-client" module, the protocol-accessible nodes are an instance of the "netconf-client-app-grouping" discussed in Section 2.1.2.4 grouping.

* The reason for why "netconf-client-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of netconf-client-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The following example illustrates configuring a NETCONF client to initiate connections, using both the SSH and TLS transport protocols, as well as to listen for call-home connections, again using both the SSH and TLS transport protocols.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<netconf-client xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-clie\
nt">

  <!-- NETCONF servers to initiate connections to -->
```

```
     <initiate>
       <netconf-server>
         <name>corp-fw1</name>
         <endpoints>
           <endpoint>
             <name>corp-fw1.example.com</name>
             <ssh>
               <tcp-client-parameters>
                 <remote-address>corp-fw1.example.com</remote-address>
                 <keepalives>
                   <idle-time>15</idle-time>
                   <max-probes>3</max-probes>
                   <probe-interval>30</probe-interval>
                 </keepalives>
               </tcp-client-parameters>
               <ssh-client-parameters>
                 <client-identity>
                   <username>foobar</username>
                   <public-key>
                     <keystore-reference>ssh-rsa-key</keystore-referenc\
   e>
                   </public-key>
                 </client-identity>
                 <server-authentication>
                   <ca-certs>
                     <truststore-reference>trusted-server-ca-certs</tru\
   ststore-reference>
                   </ca-certs>
                   <ee-certs>
                     <truststore-reference>trusted-server-ee-certs</tru\
   ststore-reference>
                   </ee-certs>
                 </server-authentication>
                 <keepalives>
                   <max-wait>30</max-wait>
                   <max-attempts>3</max-attempts>
                 </keepalives>
               </ssh-client-parameters>
               <netconf-client-parameters>
                 <!-- nothing to configure -->
               </netconf-client-parameters>
             </ssh>
           </endpoint>
           <endpoint>
             <name>corp-fw2.example.com</name>
             <tls>
               <tcp-client-parameters>
                 <remote-address>corp-fw2.example.com</remote-address>
```

```
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <netconf-client-parameters>
                <!-- nothing to configure -->
              </netconf-client-parameters>
            </tls>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
        </reconnect-strategy>
      </netconf-server>
    </initiate>
```

```
     <!-- endpoints to listen for NETCONF Call Home connections on -->
     <listen>
       <endpoint>
         <name>Intranet-facing SSH listener</name>
         <ssh>
           <tcp-server-parameters>
             <local-address>192.0.2.7</local-address>
           </tcp-server-parameters>
           <ssh-client-parameters>
             <client-identity>
               <username>foobar</username>
               <public-key>
                 <keystore-reference>ssh-rsa-key</keystore-reference>
               </public-key>
             </client-identity>
             <server-authentication>
               <ca-certs>
                 <truststore-reference>trusted-server-ca-certs</truststo\
   re-reference>
               </ca-certs>
               <ee-certs>
                 <truststore-reference>trusted-server-ee-certs</truststo\
   re-reference>
               </ee-certs>
               <ssh-host-keys>
                 <truststore-reference>trusted-ssh-public-keys</truststo\
   re-reference>
               </ssh-host-keys>
             </server-authentication>
           </ssh-client-parameters>
           <netconf-client-parameters>
             <!-- nothing to configure -->
           </netconf-client-parameters>
         </ssh>
       </endpoint>
       <endpoint>
         <name>Intranet-facing TLS listener</name>
         <tls>
           <tcp-server-parameters>
             <local-address>192.0.2.7</local-address>
           </tcp-server-parameters>
           <tls-client-parameters>
             <client-identity>
               <certificate>
                 <keystore-reference>
                   <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                   <certificate>ex-rsa-cert</certificate>
                 </keystore-reference>
```

```
              </certificate>
            </client-identity>
            <server-authentication>
              <ca-certs>
                <truststore-reference>trusted-server-ca-certs</truststo\
re-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-server-ee-certs</truststo\
re-reference>
              </ee-certs>
            </server-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-client-parameters>
          <netconf-client-parameters>
            <!-- nothing to configure -->
          </netconf-client-parameters>
        </tls>
      </endpoint>
    </listen>
  </netconf-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7589], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

<CODE BEGINS> file "ietf-netconf-client@2022-03-07.yang"

```
module ietf-netconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix ncc;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
```

```
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-ssh-client {
    prefix sshc;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
     WG List:  NETCONF WG list <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
     Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module contains a collection of YANG definitions
     for configuring NETCONF clients.

     Copyright (c) 2021 IETF Trust and the persons identified
     as authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Revised
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC HHHH
     (https://www.rfc-editor.org/info/rfcHHHH); see the RFC
     itself for full legal notices.
```

```
        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC HHHH: NETCONF Client and Server Models";
      }

      // Features

      feature ssh-initiate {
        description
          "The 'ssh-initiate' feature indicates that the NETCONF client
           supports initiating SSH connections to NETCONF servers.";
        reference
          "RFC 6242:
             Using the NETCONF Protocol over Secure Shell (SSH)";
      }

      feature tls-initiate {
        description
          "The 'tls-initiate' feature indicates that the NETCONF client
           supports initiating TLS connections to NETCONF servers.";
        reference
          "RFC 7589: Using the NETCONF Protocol over Transport
             Layer Security (TLS) with Mutual X.509 Authentication";
      }

      feature ssh-listen {
        description
          "The 'ssh-listen' feature indicates that the NETCONF client
           supports opening a port to listen for incoming NETCONF
           server call-home SSH connections.";
        reference
          "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
      }

      feature tls-listen {
        description
          "The 'tls-listen' feature indicates that the NETCONF client
           supports opening a port to listen for incoming NETCONF
           server call-home TLS connections.";
```

```
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    // Groupings

    grouping netconf-client-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         without any consideration for how underlying transport
         sessions are established.

         This grouping currently does not define any nodes.";
    }

    grouping netconf-client-initiate-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         'initiate' protocol stack for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-initiate";
          container ssh {
            description
              "Specifies IP and SSH specific configuration
               for the connection.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "830";
                  description
                    "The NETCONF client will attempt to connect
                     to the IANA-assigned well-known port value
                     for 'netconf-ssh' (830) if no value is
                     specified.";
                }
              }
            }
            container ssh-client-parameters {
              description
                "A wrapper around the SSH client parameters to
                 avoid name collisions.";
```

```
            uses sshc:ssh-client-grouping;
          }
          container netconf-client-parameters {
            description
              "A wrapper around the NETCONF client parameters
               to avoid name collisions.";
            uses ncc:netconf-client-grouping;
          }
        }
      }
      case tls {
        if-feature "tls-initiate";
        container tls {
          description
            "Specifies IP and TLS specific configuration
             for the connection.";
          container tcp-client-parameters {
            description
              "A wrapper around the TCP client parameters
               to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
              refine "remote-port" {
                default "6513";
                description
                  "The NETCONF client will attempt to connect
                   to the IANA-assigned well-known port value
                   for 'netconf-tls' (6513) if no value is
                   specified.";
              }
            }
          }
          container tls-client-parameters {
            must client-identity {
              description
                "NETCONF/TLS clients MUST pass some
                 authentication credentials.";
            }
            description
              "A wrapper around the TLS client parameters
               to avoid name collisions.";
            uses tlsc:tls-client-grouping;
          }
          container netconf-client-parameters {
            description
              "A wrapper around the NETCONF client parameters
               to avoid name collisions.";
            uses ncc:netconf-client-grouping;
          }
```

```
          }
        }
      }
    } // netconf-client-initiate-stack-grouping

    grouping netconf-client-listen-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         'listen' protocol stack for a single connection.  The
         'listen' stack supports call home connections, as
         described in RFC 8071";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-listen";
          container ssh {
            description
              "SSH-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4334";
                  description
                    "The NETCONF client will listen on the IANA-
                     assigned well-known port for 'netconf-ch-ssh'
                     (4334) if no value is specified.";
                }
              }
            }
            container ssh-client-parameters {
              description
                "A wrapper around the SSH client parameters
                 to avoid name collisions.";
              uses sshc:ssh-client-grouping;
            }
            container netconf-client-parameters {
              description
                "A wrapper around the NETCONF client parameters
                 to avoid name collisions.";
              uses ncc:netconf-client-grouping;
```

```
            }
          }
        }
        case tls {
          if-feature "tls-listen";
          container tls {
            description
              "TLS-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4334";
                  description
                    "The NETCONF client will listen on the IANA-
                      assigned well-known port for 'netconf-ch-ssh'
                      (4334) if no value is specified.";
                }
              }
            }
            container tls-client-parameters {
              must client-identity {
                description
                  "NETCONF/TLS clients MUST pass some
                   authentication credentials.";
              }
              description
                "A wrapper around the TLS client parameters
                 to avoid name collisions.";
              uses tlsc:tls-client-grouping;
            }
            container netconf-client-parameters {
              description
                "A wrapper around the NETCONF client parameters
                 to avoid name collisions.";
              uses ncc:netconf-client-grouping;
            }
          }
        }
      }
    } // netconf-client-listen-stack-grouping

    grouping netconf-client-app-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
```

```
          application that supports both 'initiate' and 'listen'
          protocol stacks for a multiplicity of connections.";
        container initiate {
          if-feature "ssh-initiate or tls-initiate";
          presence
            "Indicates that client-initiated connections have been
             configured.  This statement is present so the mandatory
             descendant nodes do not imply that this node must be
             configured.";
          description
            "Configures client initiating underlying TCP connections.";
          list netconf-server {
            key "name";
            min-elements 1;
            description
              "List of NETCONF servers the NETCONF client is to
               maintain simultaneous connections with.";
            leaf name {
              type string;
              description
                "An arbitrary name for the NETCONF server.";
            }
            container endpoints {
              description
                "Container for the list of endpoints.";
              list endpoint {
                key "name";
                min-elements 1;
                ordered-by user;
                description
                  "A user-ordered list of endpoints that the NETCONF
                   client will attempt to connect to in the specified
                   sequence.  Defining more than one enables
                   high-availability.";
                leaf name {
                  type string;
                  description
                    "An arbitrary name for the endpoint.";
                }
                uses netconf-client-initiate-stack-grouping;
              } // list endpoint
            } // container endpoints

            container connection-type {
              description
                "Indicates the NETCONF client's preference for how the
                 NETCONF connection is maintained.";
              choice connection-type {
```

```
                mandatory true;
                description
                  "Selects between available connection types.";
                case persistent-connection {
                  container persistent {
                    presence
                      "Indicates that a persistent connection is to be
                       maintained.";
                    description
                      "Maintain a persistent connection to the NETCONF
                       server.  If the connection goes down, immediately
                       start trying to reconnect to the NETCONF server,
                       using the reconnection strategy.

                       This connection type minimizes any NETCONF server
                       to NETCONF client data-transfer delay, albeit at
                       the expense of holding resources longer.";
                  }
                }
                case periodic-connection {
                  container periodic {
                    presence "Indicates that a periodic connection is
                              to be maintained.";
                    description
                      "Periodically connect to the NETCONF server.

                       This connection type increases resource
                       utilization, albeit with increased delay in
                       NETCONF server to NETCONF client interactions.

                       The NETCONF client should close the underlying
                       TCP connection upon completing planned activities.

                       In the case that the previous connection is still
                       active, establishing a new connection is NOT
                       RECOMMENDED.";
                    leaf period {
                      type uint16;
                      units "minutes";
                      default "60";
                      description
                        "Duration of time between periodic connections.";
                    }
                    leaf anchor-time {
                      type yang:date-and-time {
                        // constrained to minute-level granularity
                        pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                              + '(Z|[\+\-]\d{2}:\d{2})';
```

```
                    }
                    description
                      "Designates a timestamp before or after which a
                       series of periodic connections are determined.
                       The periodic connections occur at a whole
                       multiple interval from the anchor time.  For
                       example, for an anchor time is 15 minutes past
                       midnight and a period interval of 24 hours, then
                       a periodic connection will occur 15 minutes past
                       midnight everyday.";
                  }
                  leaf idle-timeout {
                    type uint16;
                    units "seconds";
                    default 120; // two minutes
                    description
                      "Specifies the maximum number of seconds that
                       a NETCONF session may remain idle. A NETCONF
                       session will be dropped if it is idle for an
                       interval longer then this number of seconds.
                       If set to zero, then the NETCONF client will
                       never drop a session because it is idle.";
                  }
                }
              }
            }
          }
          container reconnect-strategy {
            description
              "The reconnection strategy directs how a NETCONF client
               reconnects to a NETCONF server, after discovering its
               connection to the server has dropped, even if due to a
               reboot.  The NETCONF client starts with the specified
               endpoint and tries to connect to it max-attempts times
               before trying the next endpoint in the list (round
               robin).";
            leaf start-with {
              type enumeration {
                enum first-listed {
                  description
                    "Indicates that reconnections should start with
                     the first endpoint listed.";
                }
                enum last-connected {
                  description
                    "Indicates that reconnections should start with
                     the endpoint last connected to.  If no previous
                     connection has ever been established, then the
```

```
                    first endpoint configured is used.   NETCONF
                    clients SHOULD be able to remember the last
                    endpoint connected to across reboots.";
                }
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default "first-listed";
              description
                "Specifies which of the NETCONF server's endpoints
                 the NETCONF client should start with when trying
                 to connect to the NETCONF server.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the NETCONF client tries
                 to connect to a specific endpoint before moving on
                 to the next endpoint in the list (round robin).";
            }
          }
        } // netconf-server
      } // initiate

      container listen {
        if-feature "ssh-listen or tls-listen";
        presence
          "Indicates that client-listening ports have been configured.
           This statement is present so the mandatory descendant nodes
           do not imply that this node must be configured.";
        description
          "Configures the client to accept call-home TCP connections.";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "3600"; // one hour
          description
            "Specifies the maximum number of seconds that a NETCONF
             session may remain idle. A NETCONF session will be
             dropped if it is idle for an interval longer than this
             number of seconds.  If set to zero, then the server
             will never drop a session because it is idle.  Sessions
```

```
              that have a notification subscription active are never
              dropped.";
          }
          list endpoint {
            key "name";
            min-elements 1;
            description
              "List of endpoints to listen for NETCONF connections.";
            leaf name {
              type string;
              description
                "An arbitrary name for the NETCONF listen endpoint.";
            }
            uses netconf-client-listen-stack-grouping;
          } // endpoint
        } // listen
      } // netconf-client-app-grouping

      // Protocol accessible node for clients that implement this module.
      container netconf-client {
        uses netconf-client-app-grouping;
        description
          "Top-level container for NETCONF client configuration.";
      }
    }

    <CODE ENDS>
```

3. The "ietf-netconf-server" Module

The NETCONF server model presented in this section supports both listening for connections as well as initiating call-home connections, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the NETCONF server supports.

3.1. Data Model Overview

This section provides an overview of the "ietf-netconf-server" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-netconf-server" module:

```
Features:
  +-- ssh-listen
  +-- tls-listen
  +-- ssh-call-home
  +-- tls-call-home
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

3.1.2. Groupings

The "ietf-netconf-server" module defines the following "grouping" statements:

* netconf-server-grouping
* netconf-server-listen-stack-grouping
* netconf-server-callhome-stack-grouping
* netconf-server-app-grouping

Each of these groupings are presented in the following subsections.

3.1.2.1. The "netconf-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-grouping" grouping:

```
  grouping netconf-server-grouping
    +-- client-identity-mappings
       +---u x509c2n:cert-to-name
```

Comments:

* The "netconf-server-grouping" defines the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).

* The "client-identity-mappings" node, which must be enabled by "feature" statements, defines a mapping from certificate fields to NETCONF user names.

* For the referenced grouping statement(s):

  - The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

3.1.2.2. The "netconf-server-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-listen-stack-grouping" grouping:

```
  grouping netconf-server-listen-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-listen}?
       |  +-- ssh
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- ssh-server-parameters
       |     |  +---u sshs:ssh-server-grouping
       |     +-- netconf-server-parameters
       |        +---u ncs:netconf-server-grouping
       +--:(tls) {tls-listen}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- netconf-server-parameters
                +---u ncs:netconf-server-grouping
```

Comments:

* The "netconf-server-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack for NETCONF servers that listen for standard connections from NETCONF clients, as opposed to initiating call-home [RFC8071] connections.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.3. The "netconf-server-callhome-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-callhome-stack-grouping" grouping:

```
  grouping netconf-server-callhome-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-call-home}?
       |  +-- ssh
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- ssh-server-parameters
       |     |  +---u sshs:ssh-server-grouping
       |     +-- netconf-server-parameters
       |        +---u ncs:netconf-server-grouping
       +--:(tls) {tls-call-home}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- netconf-server-parameters
                +---u ncs:netconf-server-grouping
```

Comments:

* The "netconf-server-callhome-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF servers that initiate call-home [RFC8071] connections to NETCONF clients.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.4. The "netconf-server-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-app-grouping" grouping:

```
  grouping netconf-server-app-grouping
    +-- listen! {ssh-listen or tls-listen}?
    |  +-- idle-timeout?   uint16
    |  +-- endpoint* [name]
    |     +-- name?                                      string
    |     +---u netconf-server-listen-stack-grouping
    +-- call-home! {ssh-call-home or tls-call-home}?
       +-- netconf-client* [name]
          +-- name?                 string
          +-- endpoints
          |  +-- endpoint* [name]
          |     +-- name?                                        string
          |     +---u netconf-server-callhome-stack-grouping
          +-- connection-type
          |  +-- (connection-type)
          |     +--:(persistent-connection)
          |     |  +-- persistent!
          |     +--:(periodic-connection)
          |        +-- periodic!
          |           +-- period?         uint16
          |           +-- anchor-time?    yang:date-and-time
          |           +-- idle-timeout?   uint16
          +-- reconnect-strategy
             +-- start-with?     enumeration
             +-- max-attempts?   uint8
```

Comments:

* The "netconf-server-app-grouping" defines the configuration for a NETCONF server that supports both listening for connections from NETCONF clients as well as initiating call-home connections to NETCONF clients.

* Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "netconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
  - The "netconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-netconf-server" module:

```
module: ietf-netconf-server
  +--rw netconf-server
     +---u netconf-server-app-grouping
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

* For the "ietf-netconf-server" module, the protocol-accessible nodes are an instance of the "netconf-server-app-grouping" discussed in Section 3.1.2.4 grouping.

* The reason for why "netconf-server-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of netconf-server-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

3.2. Example Usage

The following example illustrates configuring a NETCONF server to listen for NETCONF client connections using both the SSH and TLS transport protocols, as well as configuring call-home to two NETCONF clients, one using SSH and the other using TLS.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-serv\
er">

  <!-- endpoints to listen for NETCONF connections on -->
  <listen>
    <endpoint> <!-- listening for SSH connections -->
      <name>netconf/ssh</name>
      <ssh>
```

```
          <tcp-server-parameters>
            <local-address>192.0.2.7</local-address>
          </tcp-server-parameters>
          <ssh-server-parameters>
            <server-identity>
              <host-key>
                <name>deployment-specific-certificate</name>
                <public-key>
                  <keystore-reference>ssh-rsa-key</keystore-reference>
                </public-key>
              </host-key>
            </server-identity>
            <client-authentication>
            </client-authentication>
          </ssh-server-parameters>
          <netconf-server-parameters>
            <!-- nothing to configure -->
          </netconf-server-parameters>
        </ssh>
      </endpoint>
      <endpoint> <!-- listening for TLS sessions -->
        <name>netconf/tls</name>
        <tls>
          <tcp-server-parameters>
            <local-address>192.0.2.7</local-address>
          </tcp-server-parameters>
          <tls-server-parameters>
            <server-identity>
              <certificate>
                <keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </keystore-reference>
              </certificate>
            </server-identity>
            <client-authentication>
              <ca-certs>
                <truststore-reference>trusted-client-ca-certs</truststore-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-client-ee-certs</truststore-reference>
              </ee-certs>
            </client-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
```

```
          </tls-server-parameters>
          <netconf-server-parameters>
            <client-identity-mappings>
              <cert-to-name>
                <id>1</id>
                <fingerprint>11:0A:05:11:00</fingerprint>
                <map-type>x509c2n:specified</map-type>
                <name>scooby-doo</name>
              </cert-to-name>
              <cert-to-name>
                <id>2</id>
                <map-type>x509c2n:san-any</map-type>
              </cert-to-name>
            </client-identity-mappings>
          </netconf-server-parameters>
        </tls>
      </endpoint>
    </listen>

    <!-- calling home to SSH and TLS based NETCONF clients -->
    <call-home>
      <netconf-client> <!-- SSH-based client -->
        <name>config-mgr</name>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <ssh>
              <tcp-client-parameters>
                <remote-address>east.config-mgr.example.com</remote-ad\
dress>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <ssh-server-parameters>
                <server-identity>
                  <host-key>
                    <name>deployment-specific-certificate</name>
                    <public-key>
                      <keystore-reference>ssh-rsa-key</keystore-refere\
nce>
                    </public-key>
                  </host-key>
                </server-identity>
              </ssh-server-parameters>
              <netconf-server-parameters>
```

```
                <!-- nothing to configure -->
              </netconf-server-parameters>
            </ssh>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <ssh>
              <tcp-client-parameters>
                <remote-address>west.config-mgr.example.com</remote-ad\
dress>
              </tcp-client-parameters>
              <ssh-server-parameters>
                <server-identity>
                  <host-key>
                    <name>deployment-specific-certificate</name>
                    <public-key>
                      <keystore-reference>ssh-rsa-key</keystore-refere\
nce>
                    </public-key>
                  </host-key>
                </server-identity>
              </ssh-server-parameters>
              <netconf-server-parameters>
                <!-- nothing to configure -->
              </netconf-server-parameters>
            </ssh>
          </endpoint>
        </endpoints>
        <connection-type>
          <periodic>
            <idle-timeout>300</idle-timeout>
            <period>60</period>
          </periodic>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </netconf-client>
      <netconf-client> <!-- TLS-based client -->
        <name>data-collector</name>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <tls>
              <tcp-client-parameters>
                <remote-address>east.analytics.example.com</remote-add\
ress>
```

```
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <netconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </netconf-server-parameters>
            </tls>
```

```
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <tls>
              <tcp-client-parameters>
                <remote-address>west.analytics.example.com</remote-add\
ress>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <netconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
```

```
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </netconf-server-parameters>
            </tls>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
        <reconnect-strategy>
          <start-with>first-listed</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </netconf-client>
    </call-home>
  </netconf-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7407], [RFC7589], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

<CODE BEGINS> file "ietf-netconf-server@2022-03-07.yang"

```
module ietf-netconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix ncs;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-x509-cert-to-name {
    prefix x509c2n;
    reference
      "RFC 7407: A YANG Data Model for SNMP Configuration";
  }

  import ietf-tcp-client {
```

```
      prefix tcpc;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    import ietf-tcp-server {
      prefix tcps;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    import ietf-ssh-common {
      prefix sshcmn;
      revision-date 2022-03-07; // stable grouping definitions
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    import ietf-ssh-server {
      prefix sshs;
      revision-date 2022-03-07; // stable grouping definitions
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    import ietf-tls-server {
      prefix tlss;
      revision-date 2022-03-07; // stable grouping definitions
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group";

    contact
      "WG Web:   https://datatracker.ietf.org/wg/netconf
       WG List:  NETCONF WG list <mailto:netconf@ietf.org>
       Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
       Author:   Gary Wu <mailto:garywu@cisco.com>
       Author:   Juergen Schoenwaelder
                 <mailto:j.schoenwaelder@jacobs-university.de>";

    description
      "This module contains a collection of YANG definitions
       for configuring NETCONF servers.

       Copyright (c) 2021 IETF Trust and the persons identified
```

```
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC HHHH
        (https://www.rfc-editor.org/info/rfcHHHH); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC HHHH: NETCONF Client and Server Models";
      }

      // Features

      feature ssh-listen {
        description
          "The 'ssh-listen' feature indicates that the NETCONF server
           supports opening a port to accept NETCONF over SSH
           client connections.";
        reference
          "RFC 6242:
             Using the NETCONF Protocol over Secure Shell (SSH)";
      }

      feature tls-listen {
        description
          "The 'tls-listen' feature indicates that the NETCONF server
           supports opening a port to accept NETCONF over TLS
           client connections.";
        reference
          "RFC 7589: Using the NETCONF Protocol over Transport
                     Layer Security (TLS) with Mutual X.509
                     Authentication";
```

```
    }

    feature ssh-call-home {
      description
        "The 'ssh-call-home' feature indicates that the NETCONF
         server supports initiating a NETCONF over SSH call
         home connection to NETCONF clients.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature tls-call-home {
      description
        "The 'tls-call-home' feature indicates that the NETCONF
         server supports initiating a NETCONF over TLS call
         home connection to NETCONF clients.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    // Groupings

    grouping netconf-server-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         without any consideration for how underlying transport
         sessions are established.

         Note that this grouping uses a fairly typical descendant
         node name such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue by wrapping the 'uses'
         statement in a container called, e.g.,
         'netconf-server-parameters'.  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container client-identity-mappings {
        description
          "Specifies mappings through which NETCONF client X.509
           certificates are used to determine a NETCONF username,
           per RFC 7407.

           For TLS-based transports, if no matching and valid
           cert-to-name list entry can be found, then the NETCONF
           server MUST close the connection, and MUST NOT accept
           NETCONF messages over it, per Section 7 in RFC 7589.
```

```
           For SSH-based transports, a matching cert-to-name
           entry overrides the username provided by the SSH
           implementation, consistent with the second paragraph
           of Section 3 in RFC 6242.";
        reference
          "RFC 6242:
             Using the NETCONF Protocol over Secure Shell (SSH)
           RFC 7589:
             Using the NETCONF Protocol over Transport Layer
             Security (TLS) with Mutual X.509 Authentication";
        uses x509c2n:cert-to-name {
          refine "cert-to-name/fingerprint" {
            mandatory false;
            description
              "A 'fingerprint' value does not need to be specified
               when the 'cert-to-name' mapping is independent of
               fingerprint matching.  A 'cert-to-name' having no
               fingerprint value will match any client certificate
               and therefore should only be present at the end of
               the user-ordered 'cert-to-name' list.";
          }
        }
      }
    }

    grouping netconf-server-listen-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         'listen' protocol stack for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-listen";
          container ssh {
            description
              "SSH-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "830";
                  description
                    "The NETCONF server will listen on the
```

```
                    IANA-assigned well-known port value
                    for 'netconf-ssh' (830) if no value
                    is specified.";
                }
              }
            }
            container ssh-server-parameters {
              description
                "A wrapper around the SSH server parameters
                 to avoid name collisions.";
              uses sshs:ssh-server-grouping;
            }
            container netconf-server-parameters {
              description
                "A wrapper around the NETCONF server parameters
                 to avoid name collisions.";
              uses ncs:netconf-server-grouping {
                refine "client-identity-mappings" {
                  if-feature "sshcmn:ssh-x509-certs";
                  description
                    "Augments in an 'if-feature' statement
                     ensuring the 'client-identity-mappings'
                     descendant is enabled only when SSH
                     supports X.509 certificates.";
                }
                augment "client-identity-mappings" {
                  description
                    "Adds a flag indicating if a cert-to-name
                     is required.";
                  leaf mapping-required {
                    type boolean;
                    description
                      "Indicates that the cert-to-name mapping
                       is required (i.e., the SSH-level username
                       is ignored).";
                  }
                }
              }
            }
          }
          case tls {
            if-feature "tls-listen";
            container tls {
              description
                "TLS-specific listening configuration for inbound
                 connections.";
              container tcp-server-parameters {
```

```
            description
              "A wrapper around the TCP client parameters
               to avoid name collisions.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "6513";
                description
                  "The NETCONF server will listen on the
                   IANA-assigned well-known port value
                   for 'netconf-tls' (6513) if no value
                   is specified.";
              }
            }
          }
          container tls-server-parameters {
            description
              "A wrapper around the TLS server parameters to
               avoid name collisions.";
            uses tlss:tls-server-grouping {
              refine "client-authentication" {
                must 'ca-certs or ee-certs';
                description
                  "NETCONF/TLS servers MUST validate client
                   certificates.  This configures certificates
                   at the socket-level (i.e. bags), more
                   discriminating client-certificate checks
                   SHOULD be implemented by the application.";
                reference
                  "RFC 7589:
                    Using the NETCONF Protocol over Transport Layer
                    Security (TLS) with Mutual X.509 Authentication";
              }
            }
          }
          container netconf-server-parameters {
            description
              "A wrapper around the NETCONF server parameters
               to avoid name collisions.";
            uses ncs:netconf-server-grouping {
              refine "client-identity-mappings/cert-to-name" {
                min-elements 1;
                description
                  "The TLS transport requires a mapping.";
              }
            }
          }
        }
      }
```

```
      }
    }

    grouping netconf-server-callhome-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         'call-home' protocol stack, for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-call-home";
          container ssh {
            description
              "Specifies SSH-specific call-home transport
               configuration.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "4334";
                  description
                    "The NETCONF server will attempt to connect
                     to the IANA-assigned well-known port for
                     'netconf-ch-tls' (4334) if no value is
                     specified.";
                }
              }
            }
            container ssh-server-parameters {
              description
                "A wrapper around the SSH server parameters
                 to avoid name collisions.";
              uses sshs:ssh-server-grouping;
            }
            container netconf-server-parameters {
              description
                "A wrapper around the NETCONF server parameters
                 to avoid name collisions.";
              uses ncs:netconf-server-grouping {
                refine "client-identity-mappings" {
                  if-feature "sshcmn:ssh-x509-certs";
                  description
                    "Augments in an 'if-feature' statement
                     ensuring the 'client-identity-mappings'
```

```
                    descendant is enabled only when SSH
                    supports X.509 certificates.";
                }
                augment "client-identity-mappings" {
                  description
                    "Adds a flag indicating if a cert-to-name
                     is required.";
                  leaf mapping-required {
                    type boolean;
                    description
                      "Indicates that the cert-to-name mapping
                       is required (i.e., the SSH-level username
                       is ignored).";
                  }
                }
              }
            }
          }
        }
        case tls {
          if-feature "tls-call-home";
          container tls {
            description
              "Specifies TLS-specific call-home transport
               configuration.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "4335";
                  description
                    "The NETCONF server will attempt to connect
                     to the IANA-assigned well-known port for
                     'netconf-ch-tls' (4335) if no value is
                     specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "A wrapper around the TLS server parameters to
                 avoid name collisions.";
              uses tlss:tls-server-grouping {
                refine "client-authentication" {
                  must 'ca-certs or ee-certs';
                  description
```

```
                    "NETCONF/TLS servers MUST validate client
                     certificates.  This configures certificates
                     at the socket-level (i.e. bags), more
                     discriminating client-certificate checks
                     SHOULD be implemented by the application.";
                  reference
                    "RFC 7589:
                      Using the NETCONF Protocol over Transport Layer
                      Security (TLS) with Mutual X.509 Authentication";
                }
              }
            }
            container netconf-server-parameters {
              description
                "A wrapper around the NETCONF server parameters
                 to avoid name collisions.";
              uses ncs:netconf-server-grouping {
                refine "client-identity-mappings/cert-to-name" {
                  min-elements 1;
                  description
                    "The TLS transport requires a mapping.";
                }
              }
            }
          }
        }
      }
    }

    grouping netconf-server-app-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         application that supports both 'listen' and 'call-home'
         protocol stacks for a multiplicity of connections.";
      container listen {
        if-feature "ssh-listen or tls-listen";
        presence
          "Indicates that server-listening ports have been configured.
           This statement is present so the mandatory descendant
           nodes do not imply that this node must be configured.";
        description
          "Configures listen behavior";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "3600"; // one hour
          description
            "Specifies the maximum number of seconds that a NETCONF
```

```
            session may remain idle. A NETCONF session will be
            dropped if it is idle for an interval longer than this
            number of seconds.  If set to zero, then the server
            will never drop a session because it is idle.  Sessions
            that have a notification subscription active are never
            dropped.";
        }
        list endpoint {
          key "name";
          min-elements 1;
          description
            "List of endpoints to listen for NETCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the NETCONF listen endpoint.";
          }
          uses netconf-server-listen-stack-grouping;
        }
      }
      container call-home {
        if-feature "ssh-call-home or tls-call-home";
        presence
          "Indicates that server-initiated call home connections have
           been configured.  This statement is present so the mandatory
           descendant nodes do not imply that this node must be
           configured.";
        description
          "Configures the NETCONF server to initiate the underlying
           transport connection to NETCONF clients.";
        list netconf-client {
          key "name";
          min-elements 1;
          description
            "List of NETCONF clients the NETCONF server is to
             maintain simultaneous call-home connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the remote NETCONF client.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
```

```
            description
              "A non-empty user-ordered list of endpoints for this
               NETCONF server to try to connect to in sequence.
               Defining more than one enables high-availability.";
            leaf name {
              type string;
              description
                "An arbitrary name for this endpoint.";
            }
            uses netconf-server-callhome-stack-grouping;
          }
        }
        container connection-type {
          description
            "Indicates the NETCONF server's preference for how the
             NETCONF connection is maintained.";
          choice connection-type {
            mandatory true;
            description
              "Selects between available connection types.";
            case persistent-connection {
              container persistent {
                presence
                  "Indicates that a persistent connection is to be
                   maintained.";
                description
                  "Maintain a persistent connection to the NETCONF
                   client. If the connection goes down, immediately
                   start trying to reconnect to the NETCONF client,
                   using the reconnection strategy.

                   This connection type minimizes any NETCONF client
                   to NETCONF server data-transfer delay, albeit at
                   the expense of holding resources longer.";
              }
            }
            case periodic-connection {
              container periodic {
                presence "Indicates that a periodic connection is
                          to be maintained.";
                description
                  "Periodically connect to the NETCONF client.

                   This connection type increases resource
                   utilization, albeit with increased delay in
                   NETCONF client to NETCONF client interactions.

                   The NETCONF client SHOULD gracefully close the
```

```
                   connection using <close-session> upon completing
                   planned activities.  If the NETCONF session is
                   not closed gracefully, the NETCONF server MUST
                   immediately attempt to reestablish the connection.

                   In the case that the previous connection is still
                   active (i.e., the NETCONF client has not closed
                   it yet), establishing a new connection is NOT
                   RECOMMENDED.";
                 leaf period {
                   type uint16;
                   units "minutes";
                   default "60";
                   description
                     "Duration of time between periodic connections.";
                 }
                 leaf anchor-time {
                   type yang:date-and-time {
                     // constrained to minute-level granularity
                     pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                           + '(Z|[\+\-]\d{2}:\d{2})';
                   }
                   description
                     "Designates a timestamp before or after which a
                      series of periodic connections are determined.
                      The periodic connections occur at a whole
                      multiple interval from the anchor time.  For
                      example, for an anchor time is 15 minutes past
                      midnight and a period interval of 24 hours, then
                      a periodic connection will occur 15 minutes past
                      midnight everyday.";
                 }
                 leaf idle-timeout {
                   type uint16;
                   units "seconds";
                   default "120"; // two minutes
                   description
                     "Specifies the maximum number of seconds that
                      a NETCONF session may remain idle. A NETCONF
                      session will be dropped if it is idle for an
                      interval longer than this number of seconds.
                      If set to zero, then the server will never
                      drop a session because it is idle.";
                 }
               }
             } // case periodic-connection
           } // choice connection-type
         } // container connection-type
```

```
          container reconnect-strategy {
            description
              "The reconnection strategy directs how a NETCONF server
               reconnects to a NETCONF client, after discovering its
               connection to the client has dropped, even if due to a
               reboot.  The NETCONF server starts with the specified
               endpoint and tries to connect to it max-attempts times
               before trying the next endpoint in the list (round
               robin).";
            leaf start-with {
              type enumeration {
                enum first-listed {
                  description
                    "Indicates that reconnections should start with
                     the first endpoint listed.";
                }
                enum last-connected {
                  description
                    "Indicates that reconnections should start with
                     the endpoint last connected to.  If no previous
                     connection has ever been established, then the
                     first endpoint configured is used.   NETCONF
                     servers SHOULD be able to remember the last
                     endpoint connected to across reboots.";
                }
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default "first-listed";
              description
                "Specifies which of the NETCONF client's endpoints
                 the NETCONF server should start with when trying
                 to connect to the NETCONF client.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the NETCONF server tries
                 to connect to a specific endpoint before moving on
                 to the next endpoint in the list (round robin).";
            }
          } // container reconnect-strategy
```

```
          } // list netconf-client
        } // container call-home
      } // grouping netconf-server-app-grouping

      // Protocol accessible node for servers that implement this module.
      container netconf-server {
        uses netconf-server-app-grouping;
        description
          "Top-level container for NETCONF server configuration.";
      }
    }

    <CODE ENDS>
```

4. Security Considerations

4.1. The "ietf-netconf-client" YANG Module

The "ietf-netconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

4.2. The "ietf-netconf-server" YANG Module

The "ietf-netconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-netconf-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-netconf-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-client
prefix:       ncc
reference:    RFC HHHH

name:         ietf-netconf-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-server
prefix:       ncs
reference:    RFC HHHH
```

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Renamed "keychain" to "keystore".

A.2. 01 to 02

* Added to ietf-netconf-client ability to connected to a cluster of endpoints, including a reconnection-strategy.

* Added to ietf-netconf-client the ability to configure connection-type and also keep-alive strategy.

* Updated both modules to accommodate new groupings in the ssh/tls drafts.

A.3. 02 to 03

* Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

* Changed 'netconf-client' to be a grouping (not a container).

A.4. 03 to 04

* Added RFC 8174 to Requirements Language Section.

* Replaced refine statement in ietf-netconf-client to add a mandatory true.

* Added refine statement in ietf-netconf-server to add a must statement.

* Now there are containers and groupings, for both the client and server models.

A.5. 04 to 05

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated examples to inline key and certificates (no longer a leafref to keystore)

A.6. 05 to 06

* Fixed change log missing section issue.

* Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.

* Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

* Removed "idle-timeout" from "persistent" connection config.

* Added "random-selection" for reconnection-strategy's "starts-with" enum.

* Replaced "connection-type" choice default (persistent) with "mandatory true".

* Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.

* Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

* Modified examples to be compatible with new crypto-types algs

A.9. 08 to 09

* Corrected use of "mandatory true" for "address" leafs.

* Updated examples to reflect update to groupings defined in the keystore draft.

* Updated to use groupings defined in new TCP and HTTP drafts.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

* Reformatted YANG modules.

A.11. 10 to 11

* Adjusted for the top-level "demux container" added to groupings imported from other modules.

* Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Moved "expanded" tree diagrams to the Appendix.

A.12. 11 to 12

* Removed the "Design Considerations" section.

* Removed the 'must' statement limiting keepalives in periodic connections.

* Updated models and examples to reflect removal of the "demux" containers in the imported models.

* Updated the "periodic-connnection" description statements to be more like the RESTCONF draft, especially where it described dropping the underlying TCP connection.

* Updated text to better reference where certain examples come from (e.g., which Section in which draft).

* In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.

* Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

A.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

A.14. 13 to 14

* Adjusting from change in TLS client model (removing the top-level 'certificate' container), by swapping refining-in a 'mandatory true' statement with a 'must' statement outside the 'uses' statement.

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

A.15. 14 to 15

* Refactored both the client and server modules similar to how the ietf-restconf-server module was refactored in -13 of that draft, and the ietf-restconf-client grouping.

A.16. 15 to 16

* Added refinement to make "cert-to-name/fingerprint" be mandatory false.

* Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

A.17. 16 to 17

* Updated examples to include the "*-key-format" nodes.

* Updated examples to remove the "required" nodes.

* Updated examples to remove the "client-auth-defined-elsewhere" nodes.

A.18. 17 to 18

* Updated examples to reflect new "bag" addition to truststore.

A.19. 18 to 19

* Updated examples to remove the 'algorithm' nodes.

* Updated examples to reflect the new TLS keepalives structure.

* Added keepalives to the tcp-client-parameters section in the netconf-server SSH-based call-home example.

* Added a TLS-based call-home example to the netconf-client example.

* Added a "Note to Reviewers" note to first page.

A.20. 19 to 20

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Removed expanded tree diagrams that were listed in the Appendix.

* Updated the Security Considerations section.

A.21. 20 to 21

* Cleaned up titles in the IANA Considerations section

* Fixed issues found by the SecDir review of the "keystore" draft.

A.22. 21 to 22

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.23. 22 to 23

* Floated an 'if-feature' statement in a grouping down to where the grouping is used.

* Clarified 'client-identity-mappings' for both the SSH and TLS transports.

* For netconf-client, augmented-in a 'mapping-required' flag into 'client-identity-mappings' only for the SSH transport, and refined-in a 'min-elements 1' only for the TLS transport.

* Aligned modules with `pyang -f` formatting.

A.24. 23 to 24

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

A.25. 24 to 25

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)

* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balazs Kovacs, Benoit Claise, Bert Wijnen, David Lamparter, Juergen Schoenwaelder, Ladislav Lhotka, Martin Bjoerklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Sean Turner, and Tom Petch.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-25

Abstract

This document defines two YANG modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server. Both modules support the TLS transport protocol with both standard RESTCONF and RESTCONF Call Home connections.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements (note: not all may be present):

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* EEEE --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
* GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server

* HHHH --> the assigned RFC value for draft-ietf-netconf-netconf-client-server

* IIII --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Table of Contents

1. Introduction 4

1.1. Relation to other RFCs 4
1.2. Specification Language 6
1.3. Adherence to the NMDA 6
1.4. Conventions 6
2. The "ietf-restconf-client" Module 6
2.1. Data Model Overview 6
2.2. Example Usage 11
2.3. YANG Module 15
3. The "ietf-restconf-server" Module 25
3.1. Data Model Overview 25
3.2. Example Usage 30
3.3. YANG Module 34
4. Security Considerations 46
4.1. The "ietf-restconf-client" YANG Module 46
4.2. The "ietf-restconf-server" YANG Module 47
5. IANA Considerations 47
5.1. The "IETF XML" Registry 47
5.2. The "YANG Module Names" Registry 48
6. References 48
6.1. Normative References 48
6.2. Informative References 49
Appendix A. Change Log 51
A.1. 00 to 01 51
A.2. 01 to 02 51
A.3. 02 to 03 51
A.4. 03 to 04 51
A.5. 04 to 05 52
A.6. 05 to 06 52
A.7. 06 to 07 52
A.8. 07 to 08 52
A.9. 08 to 09 52
A.10. 09 to 10 53
A.11. 10 to 11 53
A.12. 11 to 12 53
A.13. 12 to 13 53
A.14. 13 to 14 54
A.15. 14 to 15 54
A.16. 15 to 16 54
A.17. 16 to 17 54
A.18. 17 to 18 55
A.19. 18 to 19 55
A.20. 19 to 20 55
A.21. 20 to 21 55
A.22. 21 to 22 55
A.23. 22 to 23 55
A.24. 23 to 24 55
A.25. 24 to 25 56
Acknowledgements 56

Author's Address 56

1. Introduction

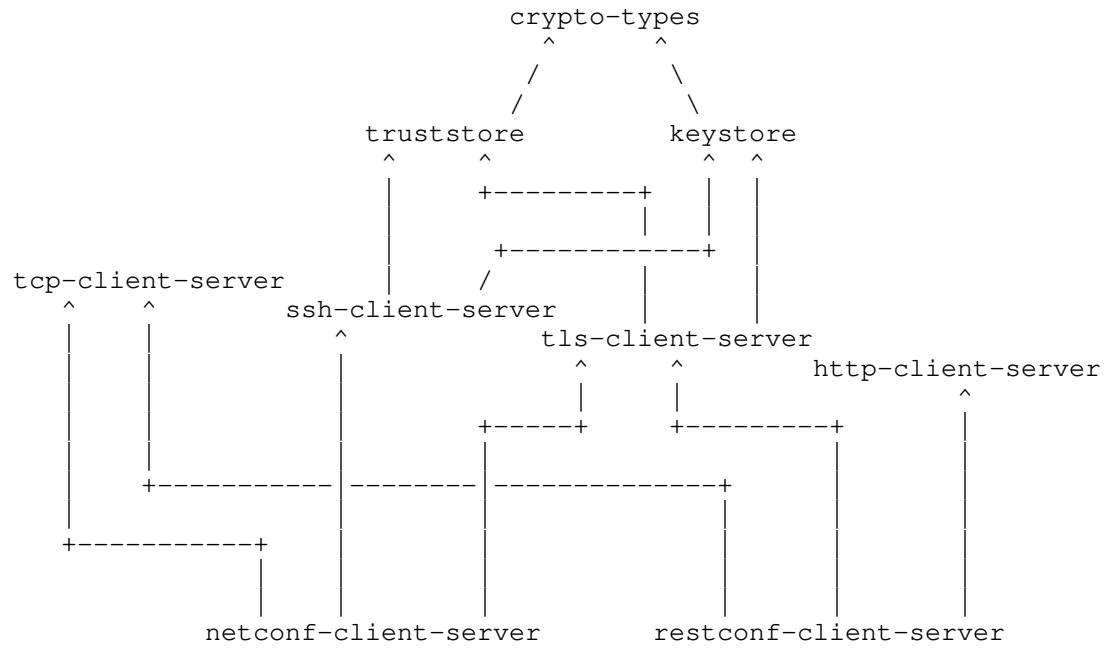
This document defines two YANG [RFC7950] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server [RFC8040]. Both modules support the TLS [RFC8446] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [RFC8071].

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-restconf-client" Module

The RESTCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the RESTCONF client supports.

2.1. Data Model Overview

This section provides an overview of the "ietf-restconf-client" module in terms of its features and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-restconf-client" module:

```
Features:
  +-- https-initiate
  +-- http-listen
  +-- https-listen
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Groupings

The "ietf-restconf-client" module defines the following "grouping" statements:

* restconf-client-grouping
* restconf-client-initiate-stack-grouping
* restconf-client-listen-stack-grouping
* restconf-client-app-grouping

Each of these groupings are presented in the following subsections.

2.1.2.1. The "restconf-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-grouping" grouping:

```
  grouping restconf-client-grouping ---> <empty>
```

Comments:

* This grouping does not define any nodes, but is maintained so that downstream modules can augment nodes into it if needed.

* The "restconf-client-grouping" defines, if it can be called that, the configuration for just "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

2.1.2.2. The "restconf-client-initiate-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-initiate-stack-grouping" grouping:

```
  grouping restconf-client-initiate-stack-grouping
    +-- (transport)
       +--:(https) {https-initiate}?
          +-- https
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
             |  +---u httpc:http-client-grouping
             +-- restconf-client-parameters
                +---u rcc:restconf-client-grouping
```

Comments:

* The "restconf-client-initiate-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that initiate connections to RESTCONF servers, as opposed to receiving call-home [RFC8071] connections.

* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.3. The "restconf-client-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-listen-stack-grouping" grouping:

```
  grouping restconf-client-listen-stack-grouping
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-client-parameters
       |     |  +---u httpc:http-client-grouping
       |     +-- restconf-client-parameters
       |        +---u rcc:restconf-client-grouping
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
             |  +---u httpc:http-client-grouping
             +-- restconf-client-parameters
                +---u rcc:restconf-client-grouping
```

Comments:

* The "restconf-client-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that receive call-home [RFC8071] connections from RESTCONF servers.

* The "transport" choice node enables both the HTTP and HTTPS transports to be configured, with each option enabled by a "feature" statement. Note that RESTCONF requires HTTPS, the HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-client.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.4. The "restconf-client-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-app-grouping" grouping:

```
  grouping restconf-client-app-grouping
    +-- initiate! {https-initiate}?
    |  +-- restconf-server* [name]
    |     +-- name?                 string
    |     +-- endpoints
    |     |  +-- endpoint* [name]
    |     |     +-- name?                                          string
    |     |     +---u restconf-client-initiate-stack-grouping
    |     +-- connection-type
    |     |  +-- (connection-type)
    |     |     +--:(persistent-connection)
    |     |     |  +-- persistent!
    |     |     +--:(periodic-connection)
    |     |        +-- periodic!
    |     |           +-- period?         uint16
    |     |           +-- anchor-time?    yang:date-and-time
    |     |           +-- idle-timeout?   uint16
    |     +-- reconnect-strategy
    |        +-- start-with?     enumeration
    |        +-- max-attempts?   uint8
    +-- listen! {http-listen or https-listen}?
       +-- idle-timeout?   uint16
       +-- endpoint* [name]
          +-- name?                                       string
          +---u restconf-client-listen-stack-grouping
```

Comments:

* The "restconf-client-app-grouping" defines the configuration for a RESTCONF client that supports both initiating connections to RESTCONF servers as well as receiving call-home connections from RESTCONF servers.

* Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "restconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.
  - The "restconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

2.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-restconf-client" module:

```
module: ietf-restconf-client
  +--rw restconf-client
     +---u restconf-client-app-grouping
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

* For the "ietf-restconf-client" module, the protocol-accessible nodes are an instance of the "restconf-client-app-grouping" discussed in Section 2.1.2.4 grouping.

* The reason for why "restconf-client-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of restconf-client-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The following example illustrates configuring a RESTCONF client to initiate connections, as well as to listen for call-home connections.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<restconf-client xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-cl\
ient">

  <!-- RESTCONF servers to initiate connections to -->
  <initiate>
    <restconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <https>
            <tcp-client-parameters>
              <remote-address>corp-fw1.example.com</remote-address>
```

```
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <client-identity>
                  <basic>
                    <user-id>bob</user-id>
                    <cleartext-password>secret</cleartext-password>
                  </basic>
                </client-identity>
              </http-client-parameters>
            </https>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <https>
              <tcp-client-parameters>
                <remote-address>corp-fw2.example.com</remote-address>
```

```
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <client-identity>
                  <basic>
                    <user-id>bob</user-id>
                    <cleartext-password>secret</cleartext-password>
                  </basic>
                </client-identity>
              </http-client-parameters>
            </https>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
      </restconf-server>
```

```
    </initiate>

    <!-- endpoints to listen for RESTCONF Call Home connections on -->
    <listen>
      <endpoint>
        <name>Intranet-facing listener</name>
        <https>
          <tcp-server-parameters>
            <local-address>11.22.33.44</local-address>
          </tcp-server-parameters>
          <tls-client-parameters>
            <client-identity>
              <certificate>
                <keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </keystore-reference>
              </certificate>
            </client-identity>
            <server-authentication>
              <ca-certs>
                <truststore-reference>trusted-server-ca-certs</truststo\
  ore-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-server-ee-certs</truststo\
  ore-reference>
              </ee-certs>
            </server-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-client-parameters>
          <http-client-parameters>
            <client-identity>
              <basic>
                <user-id>bob</user-id>
                <cleartext-password>secret</cleartext-password>
              </basic>
            </client-identity>
          </http-client-parameters>
        </https>
      </endpoint>
    </listen>
  </restconf-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC8040], and [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-tls-client-server], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-client@2022-03-07.yang"

```
module ietf-restconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-client";
  prefix rcc;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-http-client {
    prefix httpc;
    reference
      "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
```

```
       "WG Web:   https://datatracker.ietf.org/wg/netconf
        WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

     description
       "This module contains a collection of YANG definitions
        for configuring RESTCONF clients.

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC IIII
        (https://www.rfc-editor.org/info/rfcIIII); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

     revision 2022-03-07 {
       description
         "Initial version";
       reference
         "RFC IIII: RESTCONF Client and Server Models";
     }

     // Features

     feature https-initiate {
       description
         "The 'https-initiate' feature indicates that the RESTCONF
          client supports initiating HTTPS connections to RESTCONF
          servers. This feature exists as HTTPS might not be a
          mandatory to implement transport in the future.";
       reference
         "RFC 8040: RESTCONF Protocol";
     }
```

```
  feature http-listen {
    description
      "The 'https-listen' feature indicates that the RESTCONF client
       supports opening a port to listen for incoming RESTCONF
       server call-home connections.  This feature exists as not
       all RESTCONF clients may support RESTCONF call home.";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  }

  feature https-listen {
    description
      "The 'https-listen' feature indicates that the RESTCONF client
       supports opening a port to listen for incoming RESTCONF
       server call-home connections.  This feature exists as not
       all RESTCONF clients may support RESTCONF call home.";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  }

  // Groupings

  grouping restconf-client-grouping {
    description
      "A reusable grouping for configuring a RESTCONF client
       without any consideration for how underlying transport
       sessions are established.

       This grouping currently does not define any nodes.";
  }

  grouping restconf-client-initiate-stack-grouping {
    description
      "A reusable grouping for configuring a RESTCONF client
       'initiate' protocol stack for a single connection.";

    choice transport {
      mandatory true;
      description
        "Selects between available transports. This is a
         'choice' statement so as to support additional
         transport options to be augmented in.";
      case https {
        if-feature "https-initiate";
        container https {
          must 'tls-client-parameters/client-identity
                or http-client-parameters/client-identity';
          description
```

```
            "Specifies HTTPS-specific transport
             configuration.";
          container tcp-client-parameters {
            description
              "A wrapper around the TCP client parameters
               to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
              refine "remote-port" {
                default "443";
                description
                  "The RESTCONF client will attempt to
                   connect to the IANA-assigned well-known
                   port value for 'https' (443) if no value
                   is specified.";
              }
            }
          }
          container tls-client-parameters {
            description
              "A wrapper around the TLS client parameters
               to avoid name collisions.";
            uses tlsc:tls-client-grouping;
          }
          container http-client-parameters {
            description
              "A wrapper around the HTTP client parameters
               to avoid name collisions.";
            uses httpc:http-client-grouping;
          }
          container restconf-client-parameters {
            description
              "A wrapper around the HTTP client parameters
               to avoid name collisions.";
            uses rcc:restconf-client-grouping;
          }
        }
      }
    }
  } // restconf-client-initiate-stack-grouping

  grouping restconf-client-listen-stack-grouping {
    description
      "A reusable grouping for configuring a RESTCONF client
       'listen' protocol stack for a single connection.  The
       'listen' stack supports call home connections, as
       described in RFC 8071";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
```

```
      choice transport {
        mandatory true;
        description
          "Selects between available transports. This is a
           'choice' statement so as to support additional
           transport options to be augmented in.";
        case http {
          if-feature "http-listen";
          container http {
            description
              "HTTP-specific listening configuration for inbound
               connections.

               This transport option is made available to support
               deployments where the TLS connections are terminated
               by another system (e.g., a load balanacer) fronting
               the client.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4336";
                  description
                    "The RESTCONF client will listen on the IANA-
                     assigned well-known port for 'restconf-ch-tls'
                     (4336) if no value is specified.";
                }
              }
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP client parameters
                 to avoid name collisions.";
              uses httpc:http-client-grouping;
            }
            container restconf-client-parameters {
              description
                "A wrapper around the RESTCONF client parameters
                 to avoid name collisions.";
              uses rcc:restconf-client-grouping;
            }
          }
        }
        case https {
          if-feature "https-listen";
          container https {
```

```
            must 'tls-client-parameters/client-identity
                  or http-client-parameters/client-identity';
            description
              "HTTPS-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4336";
                  description
                    "The RESTCONF client will listen on the IANA-
                     assigned well-known port for 'restconf-ch-tls'
                     (4336) if no value is specified.";
                }
              }
            }
            container tls-client-parameters {
              description
                "A wrapper around the TLS client parameters
                 to avoid name collisions.";
              uses tlsc:tls-client-grouping;
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP client parameters
                 to avoid name collisions.";
              uses httpc:http-client-grouping;
            }
            container restconf-client-parameters {
              description
                "A wrapper around the RESTCONF client parameters
                 to avoid name collisions.";
              uses rcc:restconf-client-grouping;
            }
          }
        }
      }
    } // restconf-client-listen-stack-grouping

    grouping restconf-client-app-grouping {
      description
        "A reusable grouping for configuring a RESTCONF client
         application that supports both 'initiate' and 'listen'
         protocol stacks for a multiplicity of connections.";
      container initiate {
```

```
        if-feature "https-initiate";
        presence
          "Indicates that client-initiated connections have been
           configured.  This statement is present so the mandatory
           descendant nodes do not imply that this node must be
           configured.";
        description
          "Configures client initiating underlying TCP connections.";
        list restconf-server {
          key "name";
          min-elements 1;
          description
            "List of RESTCONF servers the RESTCONF client is to
             maintain simultaneous connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF server.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
              description
                "A non-empty user-ordered list of endpoints for this
                 RESTCONF client to try to connect to in sequence.
                 Defining more than one enables high-availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this endpoint.";
              }
              uses restconf-client-initiate-stack-grouping;
            }
          }
          container connection-type {
            description
              "Indicates the RESTCONF client's preference for how
               the RESTCONF connection is maintained.";
            choice connection-type {
              mandatory true;
              description
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
```

```
                    presence
                      "Indicates that a persistent connection is to be
                       maintained.";
                    description
                      "Maintain a persistent connection to the
                       RESTCONF server. If the connection goes down,
                       immediately start trying to reconnect to the
                       RESTCONF server, using the reconnection strategy.

                       This connection type minimizes any RESTCONF server
                       to RESTCONF client data-transfer delay, albeit
                       at the expense of holding resources longer.";
                  }
                }
                case periodic-connection {
                  container periodic {
                    presence
                      "Indicates that a periodic connection is to be
                       maintained.";
                    description
                      "Periodically connect to the RESTCONF server.

                       This connection type increases resource
                       utilization, albeit with increased delay
                       in RESTCONF server to RESTCONF client
                       interactions.

                       The RESTCONF client SHOULD gracefully close
                       the underlying TLS connection upon completing
                       planned activities.

                       In the case that the previous connection is
                       still active, establishing a new connection
                       is NOT RECOMMENDED.";
                    leaf period {
                      type uint16;
                      units "minutes";
                      default "60";
                      description
                        "Duration of time between periodic
                         connections.";
                    }
                    leaf anchor-time {
                      type yang:date-and-time {
                        // constrained to minute-level granularity
                        pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                              + '(Z|[\+\-]\d{2}:\d{2})';
                      }
```

```
                  description
                    "Designates a timestamp before or after which
                     a series of periodic connections are
                     determined.  The periodic connections occur
                     at a whole multiple interval from the anchor
                     time.  For example, for an anchor time is 15
                     minutes past midnight and a period interval
                     of 24 hours, then a periodic connection will
                     occur 15 minutes past midnight everyday.";
                  }
                  leaf idle-timeout {
                    type uint16;
                    units "seconds";
                    default "120"; // two minutes
                    description
                      "Specifies the maximum number of seconds
                       that the underlying TCP session may remain
                       idle. A TCP session will be dropped if it
                       is idle for an interval longer than this
                       number of seconds If set to zero, then the
                       RESTCONF client will never drop a session
                       because it is idle.";
                  }
                }
              } // periodic-connection
            } // connection-type
          } // connection-type
          container reconnect-strategy {
            description
              "The reconnection strategy directs how a RESTCONF
               client reconnects to a RESTCONF server, after
               discovering its connection to the server has
               dropped, even if due to a reboot.  The RESTCONF
               client starts with the specified endpoint and
               tries to connect to it max-attempts times before
               trying the next endpoint in the list (round
               robin).";
            leaf start-with {
              type enumeration {
                enum first-listed {
                  description
                    "Indicates that reconnections should start
                     with the first endpoint listed.";
                }
                enum last-connected {
                  description
                    "Indicates that reconnections should start
                     with the endpoint last connected to.  If
```

```
                    no previous connection has ever been
                    established, then the first endpoint
                    configured is used.   RESTCONF clients
                    SHOULD be able to remember the last
                    endpoint connected to across reboots.";
                }
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default "first-listed";
              description
                "Specifies which of the RESTCONF server's
                 endpoints the RESTCONF client should start
                 with when trying to connect to the RESTCONF
                 server.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the RESTCONF client
                 tries to connect to a specific endpoint before
                 moving on to the next endpoint in the list
                 (round robin).";
            }
          }
        }
      } // initiate
      container listen {
        if-feature "http-listen or https-listen";
        presence
          "Indicates that client-listening ports have been configured.
           This statement is present so the mandatory descendant nodes
           do not imply that this node must be configured.";
        description
          "Configures the client to accept call-home TCP connections.";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "3600"; // one hour
          description
            "Specifies the maximum number of seconds that an
             underlying TCP session may remain idle. A TCP session
```

```
              will be dropped if it is idle for an interval longer
              then this number of seconds.  If set to zero, then
              the server will never drop a session because it is
              idle.  Sessions that have a notification subscription
              active are never dropped.";
          }
          list endpoint {
            key "name";
            min-elements 1;
            description
              "List of endpoints to listen for RESTCONF connections.";
            leaf name {
              type string;
              description
                "An arbitrary name for the RESTCONF listen endpoint.";
            }
            uses restconf-client-listen-stack-grouping;
          }
        }
      } // restconf-client-app-grouping

      // Protocol accessible node for servers that implement this module.
      container restconf-client {
        uses restconf-client-app-grouping;
        description
          "Top-level container for RESTCONF client configuration.";
      }
    }
```

<CODE ENDS>

3. The "ietf-restconf-server" Module

The RESTCONF server model presented in this section supports both listening for connections as well as initiating call-home connections.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the RESTCONF server supports.

3.1. Data Model Overview

This section provides an overview of the "ietf-restconf-server" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-restconf-server" module:

```
Features:
  +-- http-listen
  +-- https-listen
  +-- https-call-home
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

3.1.2. Groupings

The "ietf-restconf-server" module defines the following "grouping" statements:

* restconf-server-grouping
* restconf-server-listen-stack-grouping
* restconf-server-callhome-stack-grouping
* restconf-server-app-grouping

Each of these groupings are presented in the following subsections.

3.1.2.1. The "restconf-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-grouping" grouping:

```
  grouping restconf-server-grouping
    +-- client-identity-mappings
       +---u x509c2n:cert-to-name
```

Comments:

* The "restconf-server-grouping" defines the configuration for just "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).

* The "client-identity-mappings" node, which must be enabled by "feature" statements, defines a mapping from certificate fields to RESTCONF user names.

* For the referenced grouping statement(s):

- The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

3.1.2.2. The "restconf-server-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-listen-stack-grouping" grouping:

```
  grouping restconf-server-listen-stack-grouping
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- external-endpoint!
       |     |  +-- address    inet:ip-address
       |     |  +-- port?      inet:port-number
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-server-parameters
       |     |  +---u https:http-server-grouping
       |     +-- restconf-server-parameters
       |        +---u rcs:restconf-server-grouping
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
             |  +---u https:http-server-grouping
             +-- restconf-server-parameters
                +---u rcs:restconf-server-grouping
```

Comments:

* The "restconf-server-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack for RESTCONF servers that listen for standard connections from RESTCONF clients, as opposed to initiating call-home [RFC8071] connections.

* The "transport" choice node enables both the HTTP and HTTPS transports to be configured, with each option enabled by a "feature" statement. The HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-server.

* For the referenced grouping statement(s):

- The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
- The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
- The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
- The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.3. The "restconf-server-callhome-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-callhome-stack-grouping" grouping:

```
  grouping restconf-server-callhome-stack-grouping
    +-- (transport)
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
             |  +---u https:http-server-grouping
             +-- restconf-server-parameters
                +---u rcs:restconf-server-grouping
```

Comments:

* The "restconf-server-callhome-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF servers that initiate call-home [RFC8071] connections to RESTCONF clients.

* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.4. The "restconf-server-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-app-grouping" grouping:

```
  grouping restconf-server-app-grouping
    +-- listen! {http-listen or https-listen}?
    |  +-- endpoint* [name]
    |     +-- name?                                        string
    |     +---u restconf-server-listen-stack-grouping
    +-- call-home! {https-call-home}?
       +-- restconf-client* [name]
          +-- name?                 string
          +-- endpoints
          |  +-- endpoint* [name]
          |     +-- name?                                          string
          |     +---u restconf-server-callhome-stack-grouping
          +-- connection-type
          |  +-- (connection-type)
          |     +--:(persistent-connection)
          |     |  +-- persistent!
          |     +--:(periodic-connection)
          |        +-- periodic!
          |           +-- period?         uint16
          |           +-- anchor-time?    yang:date-and-time
          |           +-- idle-timeout?   uint16
          +-- reconnect-strategy
             +-- start-with?     enumeration
             +-- max-attempts?   uint8
```

Comments:

* The "restconf-server-app-grouping" defines the configuration for a RESTCONF server that supports both listening for connections from RESTCONF clients as well as initiating call-home connections to RESTCONF clients.

* Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "restconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
  - The "restconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-restconf-server" module:

```
module: ietf-restconf-server
  +--rw restconf-server
     +---u restconf-server-app-grouping
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

* For the "ietf-restconf-server" module, the protocol-accessible nodes are an instance of the "restconf-server-app-grouping" discussed in Section 3.1.2.4 grouping.

* The reason for why "restconf-server-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of restconf-server-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

3.2. Example Usage

The following example illustrates configuring a RESTCONF server to listen for RESTCONF client connections, as well as configuring call-home to one RESTCONF client.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

```
================ NOTE: '\' line wrapping per RFC 8792 ================

<restconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-se\
rver">

  <!-- endpoints to listen for RESTCONF connections on -->
  <listen>
    <endpoint>
      <name>restconf/https</name>
      <https>
        <tcp-server-parameters>
          <local-address>11.22.33.44</local-address>
        </tcp-server-parameters>
        <tls-server-parameters>
```

```
            <server-identity>
              <certificate>
                <keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </keystore-reference>
              </certificate>
            </server-identity>
            <client-authentication>
              <ca-certs>
                <truststore-reference>trusted-client-ca-certs</truststo\
  ore-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-client-ee-certs</truststo\
  ore-reference>
              </ee-certs>
            </client-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-server-parameters>
          <http-server-parameters>
            <server-name>foo.example.com</server-name>
          </http-server-parameters>
          <restconf-server-parameters>
            <client-identity-mappings>
              <cert-to-name>
                <id>1</id>
                <fingerprint>11:0A:05:11:00</fingerprint>
                <map-type>x509c2n:specified</map-type>
                <name>scooby-doo</name>
              </cert-to-name>
                <cert-to-name>
                <id>2</id>
                <map-type>x509c2n:san-any</map-type>
              </cert-to-name>
            </client-identity-mappings>
          </restconf-server-parameters>
        </https>
      </endpoint>
    </listen>

    <!-- call home to a RESTCONF client with two endpoints -->
    <call-home>
      <restconf-client>
        <name>config-manager</name>
        <endpoints>
```

```
          <endpoint>
            <name>east-data-center</name>
            <https>
              <tcp-client-parameters>
                <remote-address>east.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <http-server-parameters>
                <server-name>foo.example.com</server-name>
              </http-server-parameters>
              <restconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
```

```
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </restconf-server-parameters>
            </https>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <https>
              <tcp-client-parameters>
                <remote-address>west.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <http-server-parameters>
```

```
                <server-name>foo.example.com</server-name>
              </http-server-parameters>
              <restconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </restconf-server-parameters>
            </https>
          </endpoint>
        </endpoints>
        <connection-type>
          <periodic>
            <idle-timeout>300</idle-timeout>
            <period>60</period>
          </periodic>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </restconf-client>
    </call-home>
  </restconf-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC7407], [RFC8040], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-tls-client-server], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-server@2022-03-07.yang"

```
module ietf-restconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-server";
  prefix rcs;

  import ietf-yang-types {
```

```
      prefix yang;
      reference
        "RFC 6991: Common YANG Data Types";
    }

    import ietf-inet-types {
      prefix inet;
      reference
        "RFC 6991: Common YANG Data Types";
    }

    import ietf-x509-cert-to-name {
      prefix x509c2n;
      reference
        "RFC 7407: A YANG Data Model for SNMP Configuration";
    }

    import ietf-tcp-client {
      prefix tcpc;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    import ietf-tcp-server {
      prefix tcps;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    import ietf-tls-server {
      prefix tlss;
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    import ietf-http-server {
      prefix https;
      reference
        "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group";

    contact
      "WG Web:   https://datatracker.ietf.org/wg/netconf
       WG List:  NETCONF WG list <mailto:netconf@ietf.org>
       Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
```

```
        Author:   Gary Wu <mailto:garywu@cisco.com>
        Author:   Juergen Schoenwaelder
                  <mailto:j.schoenwaelder@jacobs-university.de>";

      description
        "This module contains a collection of YANG definitions
         for configuring RESTCONF servers.

         Copyright (c) 2021 IETF Trust and the persons identified
         as authors of the code. All rights reserved.

         Redistribution and use in source and binary forms, with
         or without modification, is permitted pursuant to, and
         subject to the license terms contained in, the Revised
         BSD License set forth in Section 4.c of the IETF Trust's
         Legal Provisions Relating to IETF Documents
         (https://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC IIII
         (https://www.rfc-editor.org/info/rfcIIII); see the RFC
         itself for full legal notices.

         The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
         'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
         'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
         are to be interpreted as described in BCP 14 (RFC 2119)
         (RFC 8174) when, and only when, they appear in all
         capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC IIII: RESTCONF Client and Server Models";
      }

      // Features

      feature http-listen {
        description
          "The 'http-listen' feature indicates that the RESTCONF server
           supports opening a port to listen for incoming RESTCONF over
           TPC client connections, whereby the TLS connections are
           terminated by an external system.";
        reference
          "RFC 8040: RESTCONF Protocol";
      }
```

```
  feature https-listen {
    description
      "The 'https-listen' feature indicates that the RESTCONF server
       supports opening a port to listen for incoming RESTCONF over
       TLS client connections, whereby the TLS connections are
       terminated by the server itself.";
    reference
      "RFC 8040: RESTCONF Protocol";
  }

  feature https-call-home {
    description
      "The 'https-call-home' feature indicates that the RESTCONF
       server supports initiating connections to RESTCONF clients.";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  }

  // Groupings

  grouping restconf-server-grouping {
    description
      "A reusable grouping for configuring a RESTCONF server
       without any consideration for how underlying transport
       sessions are established.

       Note that this grouping uses a fairly typical descendant
       node name such that a stack of 'uses' statements will
       have name conflicts.  It is intended that the consuming
       data model will resolve the issue by wrapping the 'uses'
       statement in a container called, e.g.,
       'restconf-server-parameters'.  This model purposely does
       not do this itself so as to provide maximum flexibility
       to consuming models.";

    container client-identity-mappings {
      description
        "Specifies mappings through which RESTCONF client X.509
         certificates are used to determine a RESTCONF username.
         If no matching and valid cert-to-name list entry can be
         found, then the RESTCONF server MUST close the connection,
         and MUST NOT accept RESTCONF messages over it.";
      reference
        "RFC 7407: A YANG Data Model for SNMP Configuration.";
      uses x509c2n:cert-to-name {
        refine "cert-to-name/fingerprint" {
          mandatory false;
          description
```

```
              "A 'fingerprint' value does not need to be specified
               when the 'cert-to-name' mapping is independent of
               fingerprint matching.  A 'cert-to-name' having no
               fingerprint value will match any client certificate
               and therefore should only be present at the end of
               the user-ordered 'cert-to-name' list.";
          }
        }
      }
    }

    grouping restconf-server-listen-stack-grouping {
      description
        "A reusable grouping for configuring a RESTCONF server
         'listen' protocol stack for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports. This is a
           'choice' statement so as to support additional
           transport options to be augmented in.";
        case http {
          if-feature "http-listen";
          container http {
            description
              "Configures RESTCONF server stack assuming that
               TLS-termination is handled externally.";
            container external-endpoint {
              presence
                "Identifies that an external endpoint has been
                 configured.  This statement is present so the
                 mandatory descendant nodes do not imply that
                 this node must be configured.";
              description
                "Identifies contact information for the external
                 system that terminates connections before passing
                 them thru to this server (e.g., a network address
                 translator or a load balancer).  These values have
                 no effect on the local operation of this server,
                 but may be used by the application when needing to
                 inform other systems how to contact this server.";
              leaf address {
                type inet:ip-address;
                mandatory true;
                description
                  "The IP address or hostname of the external system
                   that terminates incoming RESTCONF client
                   connections before forwarding them to this
```

```
                server.";
            }
            leaf port {
              type inet:port-number;
              default "443";
              description
                "The port number that the external system listens
                 on for incoming RESTCONF client connections that
                 are forwarded to this server.  The default HTTPS
                 port (443) is used, as expected for a RESTCONF
                 connection.";
            }
          }
          container tcp-server-parameters {
            description
              "A wrapper around the TCP server parameters
               to avoid name collisions.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "80";
                description
                  "The RESTCONF server will listen on the IANA-
                   assigned well-known port value for 'http'
                   (80) if no value is specified.";
              }
            }
          }
          container http-server-parameters {
            description
              "A wrapper around the HTTP server parameters
               to avoid name collisions.";
            uses https:http-server-grouping;
          }
          container restconf-server-parameters {
            description
              "A wrapper around the RESTCONF server parameters
               to avoid name collisions.";
            uses rcs:restconf-server-grouping;
          }
        }
      }
      case https {
        if-feature "https-listen";
        container https {
          description
            "Configures RESTCONF server stack assuming that
             TLS-termination is handled internally.";
          container tcp-server-parameters {
```

```
            description
              "A wrapper around the TCP server parameters
               to avoid name collisions.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "443";
                description
                  "The RESTCONF server will listen on the IANA-
                   assigned well-known port value for 'https'
                   (443) if no value is specified.";
              }
            }
          }
          container tls-server-parameters {
            description
              "A wrapper around the TLS server parameters
               to avoid name collisions.";
            uses tlss:tls-server-grouping;
          }
          container http-server-parameters {
            description
              "A wrapper around the HTTP server parameters
               to avoid name collisions.";
            uses https:http-server-grouping;
          }
          container restconf-server-parameters {
            description
              "A wrapper around the RESTCONF server parameters
               to avoid name collisions.";
            uses rcs:restconf-server-grouping;
          }
        }
      }
    }
  }

  grouping restconf-server-callhome-stack-grouping {
    description
      "A reusable grouping for configuring a RESTCONF server
       'call-home' protocol stack, for a single connection.";
    choice transport {
      mandatory true;
      description
        "Selects between available transports. This is a
         'choice' statement so as to support additional
         transport options to be augmented in.";
      case https {
        if-feature "https-listen";
```

```
          container https {
            description
              "Configures RESTCONF server stack assuming that
               TLS-termination is handled internally.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "4336";
                  description
                    "The RESTCONF server will attempt to
                     connect to the IANA-assigned well-known
                     port for 'restconf-ch-tls' (4336) if no
                     value is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "A wrapper around the TLS server parameters
                 to avoid name collisions.";
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP server parameters
                 to avoid name collisions.";
              uses https:http-server-grouping;
            }
            container restconf-server-parameters {
              description
                "A wrapper around the RESTCONF server parameters
                 to avoid name collisions.";
              uses rcs:restconf-server-grouping;
            }
          }
        }
      }
    }

    grouping restconf-server-app-grouping {
      description
        "A reusable grouping for configuring a RESTCONF server
         application that supports both 'listen' and 'call-home'
         protocol stacks for a multiplicity of connections.";
      container listen {
```

```
        if-feature "http-listen or https-listen";
        presence
          "Identifies that the server has been configured to
           listen for incoming client connections.  This statement
           is present so the mandatory descendant nodes do not
           imply that this node must be configured.";
        description
          "Configures the RESTCONF server to listen for RESTCONF
           client connections.";
        list endpoint {
          key "name";
          min-elements 1;
          description
            "List of endpoints to listen for RESTCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF listen endpoint.";
          }
          uses restconf-server-listen-stack-grouping;
        }
      }
      container call-home {
        if-feature "https-call-home";
        presence
          "Identifies that the server has been configured to initiate
           call home connections.  This statement is present so the
           mandatory descendant nodes do not imply that this node
           must be configured.";
        description
          "Configures the RESTCONF server to initiate the underlying
           transport connection to RESTCONF clients.";
        list restconf-client {
          key "name";
          min-elements 1;
          description
            "List of RESTCONF clients the RESTCONF server is to
             maintain simultaneous call-home connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the remote RESTCONF client.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
```

```
            min-elements 1;
            ordered-by user;
            description
              "User-ordered list of endpoints for this RESTCONF
               client.  Defining more than one enables high-
               availability.";
            leaf name {
              type string;
              description
                "An arbitrary name for this endpoint.";
            }
            uses restconf-server-callhome-stack-grouping;
          }
        }
        container connection-type {
          description
            "Indicates the RESTCONF server's preference for how the
             RESTCONF connection is maintained.";
          choice connection-type {
            mandatory true;
            description
              "Selects between available connection types.";
            case persistent-connection {
              container persistent {
                presence
                  "Indicates that a persistent connection is to be
                   maintained.";
                description
                  "Maintain a persistent connection to the RESTCONF
                   client. If the connection goes down, immediately
                   start trying to reconnect to the RESTCONF server,
                   using the reconnection strategy.

                   This connection type minimizes any RESTCONF
                   client to RESTCONF server data-transfer delay,
                   albeit at the expense of holding resources
                   longer.";
              }
            }
            case periodic-connection {
              container periodic {
                presence
                  "Indicates that a periodic connection is to be
                   maintained.";
                description
                  "Periodically connect to the RESTCONF client.

                   This connection type increases resource
```

```
          utilization, albeit with increased delay in
          RESTCONF client to RESTCONF client interactions.

          The RESTCONF client SHOULD gracefully close
          the underlying TLS connection upon completing
          planned activities.  If the underlying TLS
          connection is not closed gracefully, the
          RESTCONF server MUST immediately attempt
          to reestablish the connection.

          In the case that the previous connection is
          still active (i.e., the RESTCONF client has not
          closed it yet), establishing a new connection
          is NOT RECOMMENDED.";

        leaf period {
          type uint16;
          units "minutes";
          default "60";
          description
            "Duration of time between periodic connections.";
        }
        leaf anchor-time {
          type yang:date-and-time {
            // constrained to minute-level granularity
            pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                  + '(Z|[\+\-]\d{2}:\d{2})';
          }
          description
            "Designates a timestamp before or after which a
             series of periodic connections are determined.
             The periodic connections occur at a whole
             multiple interval from the anchor time.  For
             example, for an anchor time is 15 minutes past
             midnight and a period interval of 24 hours, then
             a periodic connection will occur 15 minutes past
             midnight everyday.";
        }
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "120"; // two minutes
          description
            "Specifies the maximum number of seconds that
             the underlying TCP session may remain idle.
             A TCP session will be dropped if it is idle
             for an interval longer than this number of
             seconds.  If set to zero, then the server
```

```
                    will never drop a session because it is idle.";
                }
              }
            }
          }
        }
        container reconnect-strategy {
          description
            "The reconnection strategy directs how a RESTCONF server
             reconnects to a RESTCONF client after discovering its
             connection to the client has dropped, even if due to a
             reboot.  The RESTCONF server starts with the specified
             endpoint and tries to connect to it max-attempts times
             before trying the next endpoint in the list (round
             robin).";
          leaf start-with {
            type enumeration {
              enum first-listed {
                description
                  "Indicates that reconnections should start with
                   the first endpoint listed.";
              }
              enum last-connected {
                description
                  "Indicates that reconnections should start with
                   the endpoint last connected to.  If no previous
                   connection has ever been established, then the
                   first endpoint configured is used.   RESTCONF
                   servers SHOULD be able to remember the last
                   endpoint connected to across reboots.";
              }
              enum random-selection {
                description
                  "Indicates that reconnections should start with
                   a random endpoint.";
              }
            }
            default "first-listed";
            description
              "Specifies which of the RESTCONF client's endpoints
               the RESTCONF server should start with when trying
               to connect to the RESTCONF client.";
          }
          leaf max-attempts {
            type uint8 {
              range "1..max";
            }
            default "3";
```

```
              description
                "Specifies the number times the RESTCONF server tries
                 to connect to a specific endpoint before moving on to
                 the next endpoint in the list (round robin).";
            }
          }
        } // restconf-client
      } // call-home
    } // restconf-server-app-grouping

    // Protocol accessible node for servers that implement this module.
    container restconf-server {
      uses restconf-server-app-grouping;
      description
        "Top-level container for RESTCONF server configuration.";
    }
  }
```

<CODE ENDS>

4. Security Considerations

4.1. The "ietf-restconf-client" YANG Module

The "ietf-restconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

4.2. The "ietf-restconf-server" YANG Module

The "ietf-restconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-restconf-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-restconf-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-client
prefix:       ncc
reference:    RFC IIII

name:         ietf-restconf-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-server
prefix:       ncs
reference:    RFC IIII
```

6. References

6.1. Normative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Renamed "keychain" to "keystore".

A.2. 01 to 02

* Filled in previously missing 'ietf-restconf-client' module.

* Updated the ietf-restconf-server module to accommodate new grouping 'ietf-tls-server-grouping'.

A.3. 02 to 03

* Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

* Changed restconf-client??? to be a grouping (not a container).

A.4. 03 to 04

* Added RFC 8174 to Requirements Language Section.

* Replaced refine statement in ietf-restconf-client to add a mandatory true.

* Added refine statement in ietf-restconf-server to add a must statement.

* Now there are containers and groupings, for both the client and server models.

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated examples to inline key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Updated examples to inline key and certificates (no longer a leafref to keystore)

A.6. 05 to 06

* Fixed change log missing section issue.
* Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.
* Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

* removed "idle-timeout" from "persistent" connection config.
* Added "random-selection" for reconnection-strategy's "starts-with" enum.
* Replaced "connection-type" choice default (persistent) with "mandatory true".
* Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.
* Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

* Modified examples to be compatible with new crypto-types algs

A.9. 08 to 09

* Corrected use of "mandatory true" for "address" leafs.
* Updated examples to reflect update to groupings defined in the keystore draft.
* Updated to use groupings defined in new TCP and HTTP drafts.
* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

* Reformatted YANG modules.

A.11. 10 to 11

* Adjusted for the top-level "demux container" added to groupings imported from other modules.

* Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Moved "expanded" tree diagrams to the Appendix.

A.12. 11 to 12

* Removed the 'must' statement limiting keepalives in periodic connections.

* Updated models and examples to reflect removal of the "demux" containers in the imported models.

* Updated the "periodic-connnection" description statements to better describe behavior when connections are not closed gracefully.

* Updated text to better reference where certain examples come from (e.g., which Section in which draft).

* In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.

* Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

A.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

* In ietf-restconf-server, Added 'http-listen' (not https-listen) choice, to support case when server is behind a TLS-terminator.

* Refactored server module to be more like other 'server' models. If folks like it, will also apply to the client model, as well as to both the netconf client/server models. Now the 'restconf-server-grouping' is just the RC-specific bits (i.e., the "demux" container minus the container), 'restconf-server-[listen|callhome]-stack-grouping' is the protocol stack for a single connection, and 'restconf-server-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

A.14. 13 to 14

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

* Adjusting from change in TLS client model (removing the top-level 'certificate' container).

* Added "external-endpoint" to the "http-listen" choice in ietf-restconf-server.

A.15. 14 to 15

* Added missing "or https-listen" clause in a "must" expression.

* Refactored the client module similar to how the server module was refactored in -13. Now the 'restconf-client-grouping' is just the RC-specific bits, the 'restconf-client-[initiate|listen]-stack-grouping' is the protocol stack for a single connection, and 'restconf-client-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

A.16. 15 to 16

* Added refinement to make "cert-to-name/fingerprint" be mandatory false.

* Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

* Updated restconf-client example to reflect that http-client-grouping no longer has a "protocol-version" leaf.

A.17. 16 to 17

* Updated examples to include the "*-key-format" nodes.

* Updated examples to remove the "required" nodes.

A.18. 17 to 18

* Updated examples to reflect new "bag" addition to truststore.

A.19. 18 to 19

* Updated examples to remove the 'algorithm' nodes.

* Updated examples to reflect the new TLS keepalives structure.

* Removed the 'protocol-versions' node from the restconf-server examples.

* Added a "Note to Reviewers" note to first page.

A.20. 19 to 20

* Moved and changed "must" statement so that either TLS *or* HTTP auth must be configured.

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.21. 20 to 21

* Cleaned up titles in the IANA Consideratons section

* Fixed issues found by the SecDir review of the "keystore" draft.

A.22. 21 to 22

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.23. 22 to 23

* Further clarified why some 'presence' statements are present.

* Addressed nits found in YANG Doctor reviews.

* Aligned modules with `pyang -f` formatting.

A.24. 23 to 24

* Removed Appendix A with fully-expanded tree diagrams.

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

A.25. 24 to 25

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)

* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balazs Kovacs, Benoit Claise, Bert Wijnen David Lamparter, Juergen Schoenwaelder, Ladislav Lhotka, Martin Bjoerklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Sean Turner, and Tom Petch.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-27

Abstract

This document defines three YANG 1.1 modules: the first defines features and groupings common to both SSH clients and SSH servers, the second defines a grouping for a generic SSH client, and the third defines a grouping for a generic SSH server.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* EEEE --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix B. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 4
  1.1. Relation to other RFCs 5
  1.2. Specification Language 6
  1.3. Adherence to the NMDA 6
  1.4. Conventions 6
2. The "ietf-ssh-common" Module 7
  2.1. Data Model Overview 7
  2.2. Example Usage 8
  2.3. YANG Module 9
3. The "ietf-ssh-client" Module 15
  3.1. Data Model Overview 15
  3.2. Example Usage 17
  3.3. YANG Module 21
4. The "ietf-ssh-server" Module 28
  4.1. Data Model Overview 28

4.2. Example Usage 31
4.3. YANG Module 35
5. Security Considerations 44
5.1. The "iana-ssh-key-exchange-algs" Module 44
5.2. The "iana-ssh-encryption-algs" Module 44
5.3. The "iana-ssh-mac-algs" Module 45
5.4. The "iana-ssh-public-key-algs" Module 45
5.5. The "ietf-ssh-common" YANG Module 46
5.6. The "ietf-ssh-client" YANG Module 47
5.7. The "ietf-ssh-server" YANG Module 48
6. IANA Considerations 48
6.1. The "IETF XML" Registry 48
6.2. The "YANG Module Names" Registry 49
6.3. The "iana-ssh-encryption-algs" Module 50
6.4. The "iana-ssh-mac-algs" Module 51
6.5. The "iana-ssh-public-key-algs" Module 51
6.6. The "iana-ssh-key-exchange-algs" Module 52
7. References 52
7.1. Normative References 52
7.2. Informative References 54
Appendix A. YANG Modules for IANA 56
A.1. Initial Module for the "Encryption Algorithm Names" Registry 56
A.1.1. Data Model Overview 57
A.1.2. Example Usage 58
A.1.3. YANG Module 58
A.2. Initial Module for the "MAC Algorithm Names" Registry . . 66
A.2.1. Data Model Overview 66
A.2.2. Example Usage 67
A.2.3. YANG Module 68
A.3. Initial Module for the "Public Key Algorithm Names" Registry 71
A.3.1. Data Model Overview 71
A.3.2. Example Usage 73
A.3.3. YANG Module 73
A.4. Initial Module for the "Key Exchange Method Names" Registry 82
A.4.1. Data Model Overview 82
A.4.2. Example Usage 84
A.4.3. YANG Module 84
Appendix B. Change Log 130
B.1. 00 to 01 130
B.2. 01 to 02 131
B.3. 02 to 03 131
B.4. 03 to 04 131
B.5. 04 to 05 132
B.6. 05 to 06 132
B.7. 06 to 07 132

B.8. 07 to 08 132
B.9. 08 to 09 132
B.10. 09 to 10 133
B.11. 10 to 11 133
B.12. 11 to 12 133
B.13. 12 to 13 133
B.14. 13 to 14 133
B.15. 14 to 15 133
B.16. 15 to 16 134
B.17. 16 to 17 134
B.18. 17 to 18 134
B.19. 18 to 19 135
B.20. 19 to 20 135
B.21. 20 to 21 135
B.22. 21 to 22 136
B.23. 22 to 23 136
B.24. 23 to 24 136
B.25. 24 to 25 136
B.26. 25 to 26 137
B.27. 26 to 27 137
Acknowledgements 137
Contributors 137
Author's Address 137

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines features and groupings common to both SSH clients and SSH servers, the second defines a grouping for a generic SSH client, and the third defines a grouping for a generic SSH server. It is intended that these groupings will be used by applications using the SSH protocol [RFC4252], [RFC4253], and [RFC4254]. For instance, these groupings could be used to help define the data model for an OpenSSH [OPENSSH] server or a NETCONF over SSH [RFC6242] based server.

The client and server YANG modules in this document each define one grouping, which is focused on just SSH-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen on or connect to. This affords applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "ssh-server-grouping" grouping for the SSH parts it provides, while adding data nodes for the TCP-level call-home configuration.

The modules defined in this document use groupings defined in [I-D.ietf-netconf-keystore] enabling keys to be either locally defined or a reference to globally configured values.

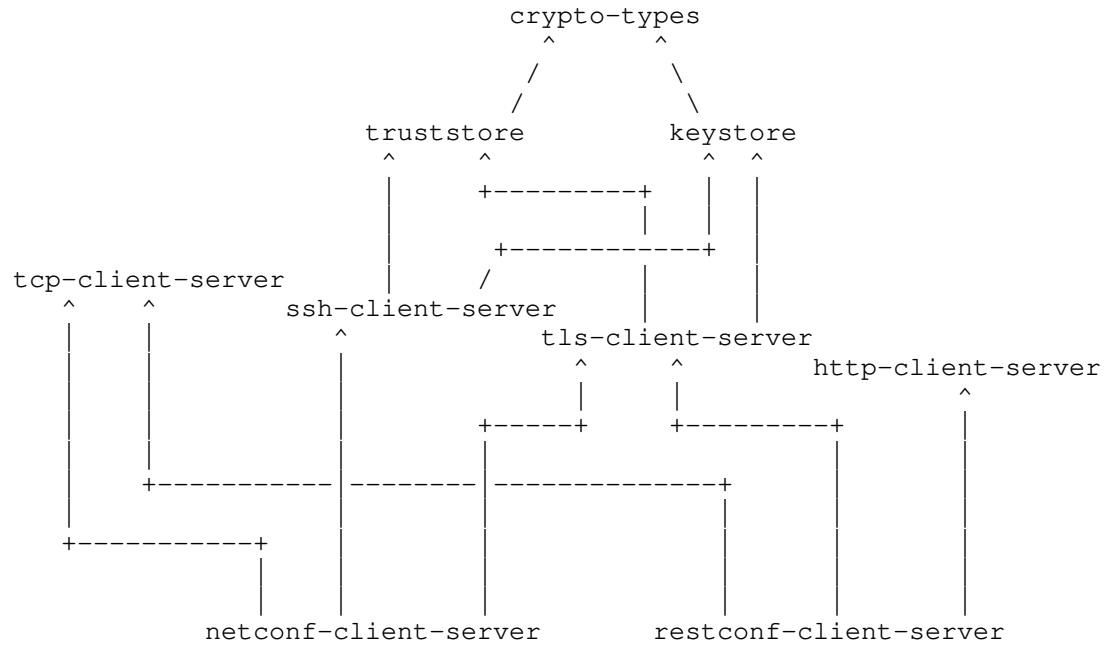
The modules defined in this document optionally support [RFC6187] enabling X.509v3 certificate based host keys and public keys.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-ssh-common" Module

The SSH common model presented in this section contains features and groupings common to both SSH clients and SSH servers. The "transport-params-grouping" grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of permitted algorithms are in decreasing order of usage preference. The algorithm that appears first in the client list that also appears in the server list is the one that is used for the SSH transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

2.1. Data Model Overview

This section provides an overview of the "ietf-ssh-common" module in terms of its features, identities, and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-ssh-common" module:

```
Features:
  +-- ssh-x509-certs
  +-- transport-params
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Groupings

The "ietf-ssh-common" module defines the following "grouping" statement:

* transport-params-grouping

This grouping is presented in the following subsection.

2.1.2.1. The "transport-params-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "transport-params-grouping" grouping:

```
  grouping transport-params-grouping
    +-- host-key
    |  +-- host-key-alg*   identityref
    +-- key-exchange
    |  +-- key-exchange-alg*   identityref
    +-- encryption
    |  +-- encryption-alg*   identityref
    +-- mac
       +-- mac-alg*   identityref
```

Comments:

* This grouping is used by both the "ssh-client-grouping" and the "ssh-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.

* This grouping enables client and server configurations to specify the algorithms that are to be used when establishing SSH sessions.

* Each list is "ordered-by user".

2.1.3. Protocol-accessible Nodes

The "ietf-ssh-common" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

2.2. Example Usage

This following example illustrates how the "transport-params-grouping' grouping appears when populated with some data.

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <transport-params
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common"
     xmlns:sshea="urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs"
     xmlns:sshkea="urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-al\
   gs"
     xmlns:sshma="urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs"
     xmlns:sshpka="urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs\
   ">
     <host-key>
       <host-key-alg>sshpka:x509v3-rsa2048-sha256</host-key-alg>
       <host-key-alg>sshpka:ssh-rsa</host-key-alg>
     </host-key>
     <key-exchange>
       <key-exchange-alg>
         sshkea:diffie-hellman-group-exchange-sha256
       </key-exchange-alg>
     </key-exchange>
     <encryption>
       <encryption-alg>sshea:aes256-ctr</encryption-alg>
       <encryption-alg>sshea:aes192-ctr</encryption-alg>
       <encryption-alg>sshea:aes128-ctr</encryption-alg>
       <encryption-alg>sshea:aes256-cbc</encryption-alg>
       <encryption-alg>sshea:aes192-cbc</encryption-alg>
       <encryption-alg>sshea:aes128-cbc</encryption-alg>
     </encryption>
     <mac>
       <mac-alg>sshma:hmac-sha2-256</mac-alg>
       <mac-alg>sshma:hmac-sha2-512</mac-alg>
     </mac>
   </transport-params>
```

2.3. YANG Module

This YANG module has normative references to [RFC4253], [RFC4344], [RFC4419], [RFC5656], [RFC6187], and [RFC6668].

```
   <CODE BEGINS> file "ietf-ssh-common@2022-03-07.yang"
```

```
   module ietf-ssh-common {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
     prefix sshcmn;

     import iana-ssh-encryption-algs {
       prefix sshea;
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }

     import iana-ssh-key-exchange-algs {
       prefix sshkea;
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }

     import iana-ssh-mac-algs {
       prefix sshma;
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }

     import iana-ssh-public-key-algs {
       prefix sshpka;
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }

     import ietf-crypto-types {
       prefix ct;
       reference
         "RFC AAAA: YANG Data Types and Groupings for Cryptography";
     }

     import ietf-keystore {
       prefix ks;
       reference
         "RFC CCCC: A YANG Data Model for a Keystore";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";

     contact
       "WG Web:   https://datatracker.ietf.org/wg/netconf
        WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
```

```
        Author:   Gary Wu <mailto:garywu@cisco.com>";

      description
        "This module defines a common features and groupings for
         Secure Shell (SSH).

         Copyright (c) 2021 IETF Trust and the persons identified
         as authors of the code. All rights reserved.

         Redistribution and use in source and binary forms, with
         or without modification, is permitted pursuant to, and
         subject to the license terms contained in, the Revised
         BSD License set forth in Section 4.c of the IETF Trust's
         Legal Provisions Relating to IETF Documents
         (https://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC EEEE
         (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
         itself for full legal notices.

         The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
         'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
         'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
         are to be interpreted as described in BCP 14 (RFC 2119)
         (RFC 8174) when, and only when, they appear in all
         capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
      }

      // Features

      feature ssh-x509-certs {
        description
          "X.509v3 certificates are supported for SSH.";
        reference
          "RFC 6187: X.509v3 Certificates for Secure Shell
                     Authentication";
      }

      feature transport-params {
        description
          "SSH transport layer parameters are configurable.";
      }
```

```
    feature public-key-generation {
      description
        "Indicates that the server implements the
         'generate-public-key' RPC.";
    }

    // Groupings

    grouping transport-params-grouping {
      description
        "A reusable grouping for SSH transport parameters.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      container host-key {
        description
          "Parameters regarding host key.";
        leaf-list host-key-alg {
          type identityref {
            base sshpka:public-key-alg-base;
          }
          ordered-by user;
          description
            "Acceptable host key algorithms in order of descending
             preference.  The configured host key algorithms should
             be compatible with the algorithm used by the configured
             private key.  Please see Section 5 of RFC EEEE for
             valid combinations.

             If this leaf-list is not configured (has zero elements)
             the acceptable host key algorithms are implementation-
             defined.";
          reference
            "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
        }
      }
      container key-exchange {
        description
          "Parameters regarding key exchange.";
        leaf-list key-exchange-alg {
          type identityref {
            base sshkea:key-exchange-alg-base;
          }
          ordered-by user;
          description
            "Acceptable key exchange algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
```

```
            the acceptable key exchange algorithms are implementation
            defined.";
        }
      }
      container encryption {
        description
          "Parameters regarding encryption.";
        leaf-list encryption-alg {
          type identityref {
            base sshea:encryption-alg-base;
          }
          ordered-by user;
          description
            "Acceptable encryption algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable encryption algorithms are implementation
             defined.";
        }
      }
      container mac {
        description
          "Parameters regarding message authentication code (MAC).";
        leaf-list mac-alg {
          type identityref {
            base sshma:mac-alg-base;
          }
          ordered-by user;
          description
            "Acceptable MAC algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable MAC algorithms are implementation-
             defined.";
        }
      }
    }

    // Protocol-accessible Nodes

    rpc generate-public-key {
      if-feature "public-key-generation";
      description
        "Requests the device to generate an public key using
         the specified key algorithm.";
      input {
```

```
        leaf algorithm {
          type sshpka:public-key-algorithm-ref;
          mandatory true;
          description
            "The algorithm to be used when generating the key.";
        }
        leaf bits {
          type uint16;
          description
            "Specifies the number of bits in the key to create.
             For RSA keys, the minimum size is 1024 bits and
             the default is 3072 bits. Generally, 3072 bits is
             considered sufficient. DSA keys must be exactly 1024
             bits as specified by FIPS 186-2.  For ECDSA keys, the
             'bits' value determines the key length by selecting
             from one of three elliptic curve sizes: 256, 384 or
             521 bits. Attempting to use bit lengths other than
             these three values for ECDSA keys will fail. ECDSA-SK,
             Ed25519 and Ed25519-SK keys have a fixed length and
             the 'bits' value, if specified, will be ignored.";
        }
        choice private-key-encoding {
          default cleartext;
          description
            "A choice amongst optional private key handling.";
          case cleartext {
            leaf cleartext {
              type empty;
              description
                "Indicates that the private key is to be returned
                 as a cleartext value.";
            }
          }
          case encrypt {
            if-feature "ct:private-key-encryption";
            container encrypt-with {
              description
                 "Indicates that the key is to be encrypted using
                  the specified symmetric or asymmetric key.";
              uses ks:encrypted-by-choice-grouping;
            }
          }
          case hide {
            if-feature "ct:hidden-keys";
            leaf hide {
              type empty;
              description
                "Indicates that the private key is to be hidden.
```

```
                  Unlike the 'cleartext' and 'encrypt' options, the
                  key returned is a placeholder for an internally
                  stored key.  See the 'Support for Built-in Keys'
                  section in RFC CCCC for information about hidden
                  keys.";
              }
            }
          }
        }
        output {
          uses ct:asymmetric-key-pair-grouping;
        }
      } // end generate-public-key

   }

   <CODE ENDS>
```

3. The "ietf-ssh-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-ssh-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-ssh-client" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-ssh-client" module:

```
Features:
  +-- ssh-client-keepalives
  +-- client-ident-password
  +-- client-ident-publickey
  +-- client-ident-hostbased
  +-- client-ident-none
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

3.1.2. Groupings

The "ietf-ssh-client" module defines the following "grouping" statement:

* ssh-client-grouping

This grouping is presented in the following subsection.

3.1.2.1. The "ssh-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "ssh-client-grouping" grouping:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

  grouping ssh-client-grouping
    +-- client-identity
    |  +-- username?      string
    |  +-- public-key! {client-ident-publickey}?
    |  |  +---u ks:local-or-keystore-asymmetric-key-grouping
    |  +-- password! {client-ident-password}?
    |  |  +---u ct:password-grouping
    |  +-- hostbased! {client-ident-hostbased}?
    |  |  +---u ks:local-or-keystore-asymmetric-key-grouping
    |  +-- none?          empty {client-ident-none}?
    |  +-- certificate! {sshcmn:ssh-x509-certs}?
    |     +---u ks:local-or-keystore-end-entity-cert-with-key-groupi\
ng
    +-- server-authentication
    |  +-- ssh-host-keys!
    |  |  +---u ts:local-or-truststore-public-keys-grouping
    |  +-- ca-certs! {sshcmn:ssh-x509-certs}?
    |  |  +---u ts:local-or-truststore-certs-grouping
    |  +-- ee-certs! {sshcmn:ssh-x509-certs}?
    |     +---u ts:local-or-truststore-certs-grouping
    +-- transport-params {sshcmn:transport-params}?
    |  +---u sshcmn:transport-params-grouping
    +-- keepalives! {ssh-client-keepalives}?
       +-- max-wait?       uint16
       +-- max-attempts?   uint8
```

Comments:

* The "client-identity" node configures a "username" and authentication methods, each enabled by a "feature" statement defined in Section 3.1.1.

* The "server-authentication" node configures trust anchors for authenticating the SSH server, with each option enabled by a "feature" statement.

* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH server. The aliveness-test occurs at the SSH protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "transport-params-grouping" grouping is discussed in Section 2.1.2.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-ssh-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "ssh-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the client identity and server authentication:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <ssh-client
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
     xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
     xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

     <!-- how this client will authenticate itself to the server -->
     <client-identity>
       <username>foobar</username>
       <public-key>
         <local-definition>
           <public-key-format>ct:ssh-public-key-format</public-key-form\
   at>
           <public-key>BASE64VALUE=</public-key>
           <private-key-format>ct:rsa-private-key-format</private-key-f\
   ormat>
           <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         </local-definition>
       </public-key>
     </client-identity>

     <!-- which host keys will this client trust -->
     <server-authentication>
       <ssh-host-keys>
         <local-definition>
           <public-key>
             <name>corp-fw1</name>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   rmat>
             <public-key>BASE64VALUE=</public-key>
           </public-key>
           <public-key>
             <name>corp-fw2</name>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   rmat>
             <public-key>BASE64VALUE=</public-key>
           </public-key>
         </local-definition>
       </ssh-host-keys>
       <ca-certs>
         <local-definition>
           <certificate>
             <name>Server Cert Issuer #1</name>
             <cert-data>BASE64VALUE=</cert-data>
```

```
          </certificate>
          <certificate>
            <name>Server Cert Issuer #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </local-definition>
      </ca-certs>
      <ee-certs>
        <local-definition>
          <certificate>
            <name>My Application #1</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>My Application #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </local-definition>
      </ee-certs>
    </server-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-client>
```

The following configuration example uses keystore-references for the client identity and truststore-references for server authentication: from the keystore:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <ssh-client
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
     xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

     <!-- how this client will authenticate itself to the server -->
     <client-identity>
       <username>foobar</username>
       <!-- can an SSH client have move than one key?
       <public-key>
         <keystore-reference>ssh-rsa-key</keystore-reference>
       </public-key>
       -->
       <certificate>
         <keystore-reference>
           <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
           <certificate>ex-rsa-cert2</certificate>
         </keystore-reference>
       </certificate>
     </client-identity>

     <!-- which host-keys will this client trust -->
     <server-authentication>
       <ssh-host-keys>
         <truststore-reference>trusted-ssh-public-keys</truststore-refe\
   rence>
       </ssh-host-keys>
       <ca-certs>
         <truststore-reference>trusted-server-ca-certs</truststore-refe\
   rence>
       </ca-certs>
       <ee-certs>
         <truststore-reference>trusted-server-ee-certs</truststore-refe\
   rence>
       </ee-certs>
     </server-authentication>

     <keepalives>
       <max-wait>30</max-wait>
       <max-attempts>3</max-attempts>
     </keepalives>

   </ssh-client>
```

3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors], and [I-D.ietf-netconf-keystore].

<CODE BEGINS> file "ietf-ssh-client@2022-03-07.yang"

```
module ietf-ssh-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
  prefix sshc;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

  import ietf-truststore {
    prefix ts;
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
```

```
        WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

      description
        "This module defines reusable groupings for SSH clients that
         can be used as a basis for specific SSH client instances.

         Copyright (c) 2021 IETF Trust and the persons identified
         as authors of the code. All rights reserved.

         Redistribution and use in source and binary forms, with
         or without modification, is permitted pursuant to, and
         subject to the license terms contained in, the Revised
         BSD License set forth in Section 4.c of the IETF Trust's
         Legal Provisions Relating to IETF Documents
         (https://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC EEEE
         (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
         itself for full legal notices.

         The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
         'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
         'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
         are to be interpreted as described in BCP 14 (RFC 2119)
         (RFC 8174) when, and only when, they appear in all
         capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
      }

      // Features

      feature ssh-client-keepalives {
        description
          "Per socket SSH keepalive parameters are configurable for
           SSH clients on the server implementing this feature.";
      }

      feature client-ident-publickey {
        description
          "Indicates that the 'publickey' authentication type, per
           RFC 4252, is supported for client identification.
```

```
          The 'publickey' authentication type is required by
          RFC 4252, but common implementations enable it to
          be disabled.";
        reference
          "RFC 4252:
           The Secure Shell (SSH) Authentication Protocol";
      }

      feature client-ident-password {
        description
          "Indicates that the 'password' authentication type, per
           RFC 4252, is supported for client identification.";
        reference
          "RFC 4252:
           The Secure Shell (SSH) Authentication Protocol";
      }

      feature client-ident-hostbased {
        description
          "Indicates that the 'hostbased' authentication type, per
           RFC 4252, is supported for client identification.";
        reference
          "RFC 4252:
           The Secure Shell (SSH) Authentication Protocol";
      }

      feature client-ident-none {
        description
          "Indicates that the 'none' authentication type, per
           RFC 4252, is supported for client identification.";
        reference
          "RFC 4252:
           The Secure Shell (SSH) Authentication Protocol";
      }

      // Groupings

      grouping ssh-client-grouping {
        description
          "A reusable grouping for configuring a SSH client without
           any consideration for how an underlying TCP session is
           established.

           Note that this grouping uses fairly typical descendant
           node names such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue (e.g., by wrapping
           the 'uses' statement in a container called
```

```
          'ssh-client-parameters').  This model purposely does
          not do this itself so as to provide maximum flexibility
          to consuming models.";

        container client-identity {
          nacm:default-deny-write;
          description
            "The username and authentication methods for the client.
             The authentication methods are unordered.  Clients may
             initially send any configured method or, per RFC 4252,
             Section 5.2, send the 'none' method to prompt the server
             to provide a list of productive methods.  Whenever a
             choice amongst methods arises, implementations SHOULD
             use a default ordering that prioritizes automation
             over human-interaction.";
          leaf username {
            type string;
            description
              "The username of this user.  This will be the username
               used, for instance, to log into an SSH server.";
          }
          container public-key {
            if-feature "client-ident-publickey";
            presence
              "Indicates that publickey-based authentication has been
               configured.  This statement is present so the mandatory
               descendant nodes do not imply that this node must be
               configured.";
            description
              "A locally-defined or referenced asymmetric key
               pair to be used for client identification.";
            reference
              "RFC CCCC: A YANG Data Model for a Keystore";
            uses ks:local-or-keystore-asymmetric-key-grouping {
              refine "local-or-keystore/local/local-definition" {
                must 'public-key-format = "ct:ssh-public-key-format"';
              }
              refine "local-or-keystore/keystore/keystore-reference" {
                must 'deref(.)/../ks:public-key-format'
                   + ' = "ct:ssh-public-key-format"';
              }
            }
          }
          container password {
            if-feature "client-ident-password";
            presence
              "Indicates that password-based authentication has been
               configured.  This statement is present so the mandatory
```

```
            descendant nodes do not imply that this node must be
            configured.";
          description
            "A password to be used to authenticate the client's
             identity.";
          uses ct:password-grouping;
        }
        container hostbased {
          if-feature "client-ident-hostbased";
          presence
            "Indicates that hostbased authentication is configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A locally-defined or referenced asymmetric key
             pair to be used for host identification.";
          reference
            "RFC CCCC: A YANG Data Model for a Keystore";
          uses ks:local-or-keystore-asymmetric-key-grouping {
            refine "local-or-keystore/local/local-definition" {
              must 'public-key-format = "ct:ssh-public-key-format"';
            }
            refine "local-or-keystore/keystore/keystore-reference" {
              must 'deref(.)/../ks:public-key-format'
                 + ' = "ct:ssh-public-key-format"';
            }
          }
        }
        leaf none {
          if-feature "client-ident-none";
          type empty;
          description
            "Indicates that 'none' algorithm is used for client
             identification.";
        }
        container certificate {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that certificate-based authentication has been
             configured.  This statement is present so the mandatory
             descendant nodes do not imply that this node must be
             configured.";
          description
            "A locally-defined or referenced certificate
             to be used for client identification.";
          reference
            "RFC CCCC: A YANG Data Model for a Keystore";
          uses ks:local-or-keystore-end-entity-cert-with-key-grouping {
```

```
            refine "local-or-keystore/local/local-definition" {
              must 'public-key-format'
                 + ' = "ct:subject-public-key-info-format"';
            }
            refine "local-or-keystore/keystore/keystore-reference"
                 + "/asymmetric-key" {
              must 'deref(.)/../ks:public-key-format'
                 + ' = "ct:subject-public-key-info-format"';
            }
          }
        }
      } // container client-identity

      container server-authentication {
        nacm:default-deny-write;
        must 'ssh-host-keys or ca-certs or ee-certs';
        description
          "Specifies how the SSH client can authenticate SSH servers.
           Any combination of authentication methods is additive and
           unordered.";
        container ssh-host-keys {
          presence
            "Indicates that the SSH host key have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A bag of SSH host keys used by the SSH client to
             authenticate SSH server host keys.  A server host key
             is authenticated if it is an exact match to a
             configured SSH host key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine
              "local-or-truststore/local/local-definition/public-key" {
              must 'public-key-format = "ct:ssh-public-key-format"';
            }
            refine
              "local-or-truststore/truststore/truststore-reference" {
              must 'deref(.)/../*/ts:public-key-format'
                 + ' = "ct:ssh-public-key-format"';
            }
          }
        }
        container ca-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that the CA certificates have been configured.
```

```
            This statement is present so the mandatory descendant
            nodes do not imply that this node must be configured.";
          description
            "A set of certificate authority (CA) certificates used by
             the SSH client to authenticate SSH servers.  A server
             is authenticated if its certificate  has a valid chain
             of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that the EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of end-entity certificates used by the SSH client
             to authenticate SSH servers.  A server is authenticated
             if its certificate is an exact match to a configured
             end-entity certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
      } // container server-authentication

      container transport-params {
        nacm:default-deny-write;
        if-feature "sshcmn:transport-params";
        description
          "Configurable parameters of the SSH transport layer.";
        uses sshcmn:transport-params-grouping;
      } // container transport-parameters

      container keepalives {
        nacm:default-deny-write;
        if-feature "ssh-client-keepalives";
        presence
          "Indicates that the SSH client proactively tests the
           aliveness of the remote SSH server.";
        description
          "Configures the keep-alive policy, to proactively test
           the aliveness of the SSH server.  An unresponsive TLS
           server is dropped after approximately max-wait *
           max-attempts seconds.  Per Section 4 of RFC 4254,
           the SSH client SHOULD send an SSH_MSG_GLOBAL_REQUEST
```

```
            message with a purposely nonexistent 'request name'
            value (e.g., keepalive@ietf.org) and the 'want reply'
            value set to '1'.";
          reference
            "RFC 4254: The Secure Shell (SSH) Connection Protocol";
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "30";
            description
              "Sets the amount of time in seconds after which if
               no data has been received from the SSH server, a
               TLS-level message will be sent to test the
               aliveness of the SSH server.";
          }
          leaf max-attempts {
            type uint8;
            default "3";
            description
              "Sets the maximum number of sequential keep-alive
               messages that can fail to obtain a response from
               the SSH server before assuming the SSH server is
               no longer alive.";
          }
        } // container keepalives
      } // grouping ssh-client-grouping

   }

   <CODE ENDS>
```

4. The "ietf-ssh-server" Module

This section defines a YANG 1.1 module called "ietf-ssh-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-ssh-server" module in terms of its features and groupings.

4.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-ssh-server" module:

```
Features:
  +-- ssh-server-keepalives
  +-- local-users-supported
  +-- local-user-auth-publickey {local-users-supported}?
  +-- local-user-auth-password {local-users-supported}?
  +-- local-user-auth-hostbased {local-users-supported}?
  +-- local-user-auth-none {local-users-supported}?
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

4.1.2. Groupings

The "ietf-ssh-server" module defines the following "grouping" statement:

* ssh-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "ssh-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "ssh-server-grouping" grouping:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

     grouping ssh-server-grouping
       +-- server-identity
       |  +-- host-key* [name]
       |     +-- name?                string
       |     +-- (host-key-type)
       |        +--:(public-key)
       |        |  +-- public-key
       |        |     +---u ks:local-or-keystore-asymmetric-key-grouping
       |        +--:(certificate)
       |           +-- certificate {sshcmn:ssh-x509-certs}?
       |              +---u ks:local-or-keystore-end-entity-cert-with-k\
   ey-grouping
       +-- client-authentication
       |  +-- users {local-users-supported}?
       |  |  +-- user* [name]
       |  |     +-- name?          string
       |  |     +-- public-keys! {local-user-auth-publickey}?
       |  |     |  +---u ts:local-or-truststore-public-keys-grouping
       |  |     +-- password?      ianach:crypt-hash
       |  |     |       {local-user-auth-password}?
       |  |     +-- hostbased! {local-user-auth-hostbased}?
       |  |     |  +---u ts:local-or-truststore-public-keys-grouping
       |  |     +-- none?          empty {local-user-auth-none}?
       |  +-- ca-certs! {sshcmn:ssh-x509-certs}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- ee-certs! {sshcmn:ssh-x509-certs}?
       |     +---u ts:local-or-truststore-certs-grouping
       +-- transport-params {sshcmn:transport-params}?
       |  +---u sshcmn:transport-params-grouping
       +-- keepalives! {ssh-server-keepalives}?
          +-- max-wait?       uint16
          +-- max-attempts?   uint8
```

Comments:

* The "server-identity" node configures the authentication methods the server can use to identify itself to clients. The ability to use a certificate is enabled by a "feature".

* The "client-authentication" node configures trust anchors for authenticating the SSH client, with each option enabled by a "feature" statement.

* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH client. The aliveness-test occurs at the SSH protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "transport-params-grouping" grouping is discussed in Section 2.1.2.1 in this document.

4.1.3. Protocol-accessible Nodes

The "ietf-ssh-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

4.2. Example Usage

This section presents two examples showing the "ssh-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the server identity and client authentication:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<ssh-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- the host-key this SSH server will present -->
```

```
     <server-identity>
       <host-key>
         <name>my-pubkey-based-host-key</name>
         <public-key>
           <local-definition>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   rmat>
             <public-key>BASE64VALUE=</public-key>
             <private-key-format>ct:rsa-private-key-format</private-key\
   -format>
             <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
           </local-definition>
         </public-key>
       </host-key>
       <host-key>
         <name>my-cert-based-host-key</name>
         <certificate>
           <local-definition>
             <public-key-format>ct:subject-public-key-info-format</publ\
   ic-key-format>
             <public-key>BASE64VALUE=</public-key>
             <private-key-format>ct:rsa-private-key-format</private-key\
   -format>
             <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
             <cert-data>BASE64VALUE=</cert-data>
           </local-definition>
         </certificate>
       </host-key>
     </server-identity>

     <!-- the client credentials this SSH server will trust -->
     <client-authentication>
       <users>
         <user>
           <name>mary</name>
           <password>$0$secret</password>
           <public-keys>
             <local-definition>
               <public-key>
                 <name>User A</name>
                 <public-key-format>ct:ssh-public-key-format</public-ke\
   y-format>
                 <public-key>BASE64VALUE=</public-key>
               </public-key>
               <public-key>
                 <name>User B</name>
                 <public-key-format>ct:ssh-public-key-format</public-ke\
   y-format>
```

```
                <public-key>BASE64VALUE=</public-key>
              </public-key>
            </local-definition>
          </public-keys>
        </user>
      </users>
      <ca-certs>
        <local-definition>
          <certificate>
            <name>Identity Cert Issuer #1</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>Identity Cert Issuer #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </local-definition>
      </ca-certs>
      <ee-certs>
        <local-definition>
          <certificate>
            <name>Application #1</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>Application #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </local-definition>
      </ee-certs>
    </client-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-server>
```

The following configuration example uses keystore-references for the server identity and truststore-references for client authentication: from the keystore:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <ssh-server
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
     xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

     <!-- the host-key this SSH server will present -->
     <server-identity>
       <host-key>
         <name>my-pubkey-based-host-key</name>
         <public-key>
           <keystore-reference>ssh-rsa-key</keystore-reference>
         </public-key>
       </host-key>
       <host-key>
         <name>my-cert-based-host-key</name>
         <certificate>
           <keystore-reference>
             <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
             <certificate>ex-rsa-cert2</certificate>
           </keystore-reference>
         </certificate>
       </host-key>
     </server-identity>

     <!-- the client credentials this SSH server will trust -->
     <client-authentication>
       <users>
         <user>
           <name>mary</name>
           <password>$0$secret</password>
           <public-keys>
             <truststore-reference>SSH Public Keys for Application A</t\
   ruststore-reference>
           </public-keys>
         </user>
       </users>
       <ca-certs>
         <truststore-reference>trusted-client-ca-certs</truststore-refe\
   rence>
       </ca-certs>
       <ee-certs>
         <truststore-reference>trusted-client-ee-certs</truststore-refe\
   rence>
       </ee-certs>
```

```
    </client-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-server>
```

4.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore] and informative references to [RFC4253] and [RFC7317].

<CODE BEGINS> file "ietf-ssh-server@2022-03-07.yang"

```
module ietf-ssh-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
  prefix sshs;

  import iana-crypt-hash {
    prefix ianach;
    reference
      "RFC 7317: A YANG Data Model for System Management";
  }

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

  import ietf-truststore {
    prefix ts;
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  import ietf-keystore {
    prefix ks;
```

```
      reference
        "RFC CCCC: A YANG Data Model for a Keystore";
    }

    import ietf-ssh-common {
      prefix sshcmn;
      revision-date 2022-03-07; // stable grouping definitions
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group";

    contact
      "WG Web:   https://datatracker.ietf.org/wg/netconf
       WG List:  NETCONF WG list <mailto:netconf@ietf.org>
       Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
       Author:   Gary Wu <mailto:garywu@cisco.com>";

    description
      "This module defines reusable groupings for SSH servers that
       can be used as a basis for specific SSH server instances.

       Copyright (c) 2021 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

       Redistribution and use in source and binary forms, with
       or without modification, is permitted pursuant to, and
       subject to the license terms contained in, the Revised
       BSD License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC EEEE
       (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
       itself for full legal notices.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
       'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
       'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
       are to be interpreted as described in BCP 14 (RFC 2119)
       (RFC 8174) when, and only when, they appear in all
       capitals, as shown here.";

    revision 2022-03-07 {
      description
        "Initial version";
```

```
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Features

    feature ssh-server-keepalives {
      description
        "Per socket SSH keepalive parameters are configurable for
         SSH servers on the server implementing this feature.";
    }

    feature local-users-supported {
      description
        "Indicates that the configuration for users can be
         configured herein, as opposed to in an application
         specific location.";
    }

    feature local-user-auth-publickey {
      if-feature "local-users-supported";
      description
        "Indicates that the 'publickey' authentication type,
         per RFC 4252, is supported for locally-defined users.

         The 'publickey' authentication type is required by
         RFC 4252, but common implementations enable it to
         be disabled.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    feature local-user-auth-password {
      if-feature "local-users-supported";
      description
        "Indicates that the 'password' authentication type,
         per RFC 4252, is supported for locally-defined users.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    feature local-user-auth-hostbased {
      if-feature "local-users-supported";
      description
        "Indicates that the 'hostbased' authentication type,
         per RFC 4252, is supported for locally-defined users.";
```

```
      reference
        "RFC 4252:
         The Secure Shell (SSH) Authentication Protocol";
    }

    feature local-user-auth-none {
      if-feature "local-users-supported";
      description
        "Indicates that the 'none' authentication type, per
         RFC 4252, is supported.  It is NOT RECOMMENDED to
         enable this feature.";
      reference
        "RFC 4252:
         The Secure Shell (SSH) Authentication Protocol";
    }

    // Groupings

    grouping ssh-server-grouping {
      description
        "A reusable grouping for configuring a SSH server without
         any consideration for how underlying TCP sessions are
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'ssh-server-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container server-identity {
        nacm:default-deny-write;
        description
          "The list of host keys the SSH server will present when
           establishing a SSH connection.";
        list host-key {
          key "name";
          min-elements 1;
          ordered-by user;
          description
            "An ordered list of host keys the SSH server will use to
             construct its ordered list of algorithms, when sending
             its SSH_MSG_KEXINIT message, as defined in Section 7.1
             of RFC 4253.";
          reference
```

```
            "RFC 4253: The Secure Shell (SSH) Transport Layer
                       Protocol";
          leaf name {
            type string;
            description
              "An arbitrary name for this host key";
          }
          choice host-key-type {
            mandatory true;
            description
              "The type of host key being specified";
            container public-key {
              description
                "A locally-defined or referenced asymmetric key pair
                 to be used for the SSH server's host key.";
              reference
                "RFC CCCC: A YANG Data Model for a Keystore";
              uses ks:local-or-keystore-asymmetric-key-grouping {
                refine "local-or-keystore/local/local-definition" {
                  must
                    'public-key-format = "ct:ssh-public-key-format"';
                }
                refine "local-or-keystore/keystore/"
                     + "keystore-reference" {
                  must 'deref(.)/../ks:public-key-format'
                     + ' = "ct:ssh-public-key-format"';
                }
              }
            }
            container certificate {
              if-feature "sshcmn:ssh-x509-certs";
              description
                "A locally-defined or referenced end-entity
                 certificate to be used for the SSH server's
                 host key.";
              reference
                "RFC CCCC: A YANG Data Model for a Keystore";
              uses
              ks:local-or-keystore-end-entity-cert-with-key-grouping {
                refine "local-or-keystore/local/local-definition" {
                  must 'public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
                }
                refine "local-or-keystore/keystore/keystore-reference"
                       + "/asymmetric-key" {
                  must 'deref(.)/../ks:public-key-format'
                       + ' = "ct:subject-public-key-info-format"';
                }
```

```
            }
          }
        }
      }
    } // container server-identity

    container client-authentication {
      nacm:default-deny-write;
      description
        "Specifies how the SSH server can authenticate SSH clients.";
      container users {
        if-feature "local-users-supported";
        description
          "A list of locally configured users.";
        list user {
          key "name";
          description
            "A locally configured user.

             The server SHOULD derive the list of authentication
             'method names' returned to the SSH client from the
             descendant nodes configured herein, per Sections
             5.1 and 5.2 in RFC 4252.

             The authentication methods are unordered.  Clients
             must authenticate to all configured methods.
             Whenever a choice amongst methods arises,
             implementations SHOULD use a default ordering
             that prioritizes automation over human-interaction.";
          leaf name {
            type string;
            description
              "The 'user name' for the SSH client, as defined in
               the SSH_MSG_USERAUTH_REQUEST message in RFC 4253.";
          }
          container public-keys {
            if-feature "local-user-auth-publickey";
            presence
              "Indicates that public keys have been configured.
               This statement is present so the mandatory descendant
               nodes do not imply that this node must be
               configured.";
            description
              "A set of SSH public keys may be used by the SSH
               server to authenticate this user.  A user is
               authenticated if its public key is an exact
               match to a configured public key.";
            reference
```

```
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine "local-or-truststore/local/local-definition"
                   + "/public-key" {
              must 'public-key-format'
                   + ' = "ct:ssh-public-key-format"';
            }
            refine "local-or-truststore/truststore/"
                   + "truststore-reference" {
              must 'deref(.)/../*/ts:public-key-format'
                   + ' = "ct:ssh-public-key-format"';
            }
          }
        }
        leaf password {
          if-feature "local-user-auth-password";
          type ianach:crypt-hash;
          description
            "The password for this user.";
        }
        container hostbased {
          if-feature "local-user-auth-hostbased";
          presence
            "Indicates that hostbased keys have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be
             configured.";
          description
            "A set of SSH host keys used by the SSH server to
             authenticate this user's host.  A user's host is
             authenticated if its host key is an exact match
             to a configured host key.";
          reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer
             RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine "local-or-truststore/local/local-definition"
                   + "/public-key" {
              must 'public-key-format'
                   + ' = "ct:ssh-public-key-format"';
            }
            refine "local-or-truststore/truststore"
                   + "/truststore-reference" {
              must 'deref(.)/../*/ts:public-key-format'
                   + ' = "ct:ssh-public-key-format"';
            }
          }
        }
```

```
            leaf none {
              if-feature "local-user-auth-none";
              type empty;
              description
                "Indicates that the 'none' method is configured
                 for this user.";
              reference
                "RFC 4252: The Secure Shell (SSH) Authentication
                           Protocol.";
            }
          }
        }
        container ca-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that CA certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply this node must be configured.";
          description
            "A set of certificate authority (CA) certificates used by
             the SSH server to authenticate SSH client certificates.
             A client certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply this node must be configured.";
          description
            "A set of client certificates (i.e., end entity
             certificates) used by the SSH server to authenticate
             the certificates presented by SSH clients.  A client
             certificate is authenticated if it is an exact match
             to a configured end-entity certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
      } // container client-authentication

      container transport-params {
        nacm:default-deny-write;
        if-feature "sshcmn:transport-params";
```

```
        description
          "Configurable parameters of the SSH transport layer.";
        uses sshcmn:transport-params-grouping;
      } // container transport-params

      container keepalives {
        nacm:default-deny-write;
        if-feature "ssh-server-keepalives";
        presence
          "Indicates that the SSH server proactively tests the
           aliveness of the remote SSH client.";
        description
          "Configures the keep-alive policy, to proactively test
           the aliveness of the SSL client.  An unresponsive SSL
           client is dropped after approximately max-wait *
           max-attempts seconds.  Per Section 4 of RFC 4254,
           the SSH server SHOULD send an SSH_MSG_GLOBAL_REQUEST
           message with a purposely nonexistent 'request name'
           value (e.g., keepalive@ietf.org) and the 'want reply'
           value set to '1'.";
        reference
          "RFC 4254: The Secure Shell (SSH) Connection Protocol";
        leaf max-wait {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default "30";
          description
            "Sets the amount of time in seconds after which
             if no data has been received from the SSL client,
             a SSL-level message will be sent to test the
             aliveness of the SSL client.";
        }
        leaf max-attempts {
          type uint8;
          default "3";
          description
            "Sets the maximum number of sequential keep-alive
             messages that can fail to obtain a response from
             the SSL client before assuming the SSL client is
             no longer alive.";
        }
      }
    } // grouping ssh-server-grouping

  }
```

<CODE ENDS>

5. Security Considerations

5.1. The "iana-ssh-key-exchange-algs" Module

The "iana-ssh-key-exchange-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. The "iana-ssh-encryption-algs" Module

The "iana-ssh-encryption-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.3. The "iana-ssh-mac-algs" Module

The "iana-ssh-mac-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.4. The "iana-ssh-public-key-algs" Module

The "iana-ssh-public-key-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.5. The "ietf-ssh-common" YANG Module

The "ietf-ssh-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since this module only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.6. The "ietf-ssh-client" YANG Module

The "ietf-ssh-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since this module only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "client-identity/password" node:

  The cleartext "password" node defined in the "ssh-client-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

  | Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.7. The "ietf-ssh-server" YANG Module

The "ietf-ssh-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since this module only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., or even the modification of transport or keepalive parameters can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers seven URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs
Registrant Contact: IANA
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs
Registrant Contact: IANA
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs
Registrant Contact: IANA
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs
Registrant Contact: IANA
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-common
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

6.2. The "YANG Module Names" Registry

This document registers seven YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
   name:         iana-ssh-key-exchange-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs
   prefix:       sshkea
   reference:    RFC EEEE

   name:         iana-ssh-encryption-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs
   prefix:       sshea
   reference:    RFC EEEE

   name:         iana-ssh-mac-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs
   prefix:       sshma
   reference:    RFC EEEE

   name:         iana-ssh-public-key-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs
   prefix:       sshpka
   reference:    RFC EEEE

   name:         ietf-ssh-common
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-common
   prefix:       sshcmn
   reference:    RFC EEEE

   name:         ietf-ssh-client
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-client
   prefix:       sshc
   reference:    RFC EEEE

   name:         ietf-ssh-server
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-server
   prefix:       sshs
   reference:    RFC EEEE
```

6.3. The "iana-ssh-encryption-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-encryption-algs" that shadows the "Encryption Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-ENC-ALGS].

This registry defines a YANG identity for each encryption algorithm, and a "base" identity from which all of the other identities are derived.

An initial version of this module can be found in Appendix A.1

* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

6.4. The "iana-ssh-mac-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-mac-algs" that shadows the "MAC Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-MAC-ALGS].

This registry defines a YANG identity for each MAC algorithm, and a "base" identity from which all of the other identities are derived.

An initial version of this module can be found in Appendix A.2.

* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

6.5. The "iana-ssh-public-key-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-public-key-algs" that shadows the "Public Key Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-PUBKEY-ALGS].

This registry defines a YANG identity for each public key algorithm, and a "base" identity from which all of the other identities are derived.

Registry entries for which the '*All values beginning with the specified string and not containing "@".' note applies MUST be expanded so that there is a distinct YANG identity for each enumeration.

An initial version of this module can be found in Appendix A.3.

* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

6.6. The "iana-ssh-key-exchange-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-key-exchange-algs" that shadows the "Key Exchange Method Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-KEYEX-ALGS].

This registry defines a YANG identity for each key exchange algortihm, and a "base" identity from which all of the other identities are derived.

Registry entries for which the '*All values beginning with the specified string and not containing "@".' note applies MUST be expanded so that there is a distinct YANG identity for each enumeration.

An initial version of this module can be found in Appendix A.4.

* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

* Please also note that the "status" statement has been set to "deprecated" https://datatracker.ietf.org/doc/html/rfc8732#section-6. It is recommended that IANA adds a column to the registry to more easily track the deprecation status of algorithms.

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <https://www.rfc-editor.org/info/rfc4344>.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <https://www.rfc-editor.org/info/rfc4419>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <https://www.rfc-editor.org/info/rfc6668>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[IANA-ENC-ALGS]
(IANA), I. A. N. A., "IANA "Encryption Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-17>.

[IANA-KEYEX-ALGS]
(IANA), I. A. N. A., "IANA "Key Exchange Method Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-16>.

[IANA-MAC-ALGS]
(IANA), I. A. N. A., "IANA "MAC Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-18>.

[IANA-PUBKEY-ALGS]
(IANA), I. A. N. A., "IANA "Public Key Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-19>.

[OPENSSH] Project, T. O., "OpenSSH", <http://www.openssh.com>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, DOI 10.17487/RFC4254, January 2006, <https://www.rfc-editor.org/info/rfc4254>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. YANG Modules for IANA

The modules contained in this section were generated by scripts using the contents of the associated sub-registry as they existed on June 1st, 2021.

A.1. Initial Module for the "Encryption Algorithm Names" Registry

A.1.1. Data Model Overview

This section provides an overview of the "iana-ssh-encryption-algs" module in terms of its identities and protocol-accessible nodes.

A.1.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-ENC-ALGS].

```
Identities:
  +-- encryption-alg-base
    +-- <identity-name from IANA registry>
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

A.1.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-encryption-algs" module:

```
Typedefs:
  identityref
    +-- encryption-algorithm-ref
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* The typedef defined in the "iana-ssh-encryption-algs" module extends the "identityref" type defined in [RFC7950].

A.1.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-encryption-algs" module:

```
module: iana-ssh-encryption-algs
  +--ro supported-algorithms
     +--ro supported-algorithm*   encryption-algorithm-ref
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

A.1.2. Example Usage

The following example illustrates operational state data indicating the SSH encryption algorithms supported by the server:

```
<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs"
  xmlns:sshea="urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs">
  <supported-algorithm>sshea:aes256-ctr</supported-algorithm>
  <supported-algorithm>sshea:aes256-cbc</supported-algorithm>
  <supported-algorithm>sshea:twofish256-cbc</supported-algorithm>
  <supported-algorithm>sshea:serpent256-cbc</supported-algorithm>
  <supported-algorithm>sshea:arcfour256</supported-algorithm>
  <supported-algorithm>sshea:serpent256-ctr</supported-algorithm>
  <supported-algorithm>sshea:aead-aes-256-gcm</supported-algorithm>
</supported-algorithms>
```

A.1.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-ssh-encryption-algs@2021-06-01.yang"

module iana-ssh-encryption-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs";
  prefix sshea;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
     Tel:    +1 310 301 5800
     Email:  iana@iana.org";

  description
    "This module defines identities for the encryption algorithms
     defined in the 'Encryption Algorithm Names' sub-registry of the
```

```
        'Secure Shell (SSH) Protocol Parameters' registry maintained
        by IANA.

        Copyright (c) 2021 IETF Trust and the persons identified as
        authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        The initial version of this YANG module is part of RFC EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
        itself for full legal notices.";

    revision 2021-06-01 {
      description
        "Initial version";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Typedefs

    typedef encryption-algorithm-ref {
      type identityref {
        base "encryption-alg-base";
      }
      description
        "A reference to a SSH encryption algorithm identifier.";
    }


    // Identities

    identity encryption-alg-base {
      description
        "Base identity used to identify encryption algorithms.";
    }

    identity triple-des-cbc { // YANG IDs cannot begin with a number
      base encryption-alg-base;
      description
        "3DES-CBC";
      reference
        "RFC 4253:
```

```
          The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity blowfish-cbc {
      base encryption-alg-base;
      description
        "BLOWFISH-CBC";
      reference
        "RFC 4253:
          The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish256-cbc {
      base encryption-alg-base;
      description
        "TWOFISH256-CBC";
      reference
        "RFC 4253:
          The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish-cbc {
      base encryption-alg-base;
      description
        "TWOFISH-CBC";
      reference
        "RFC 4253:
          The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish192-cbc {
      base encryption-alg-base;
      description
        "TWOFISH192-CBC";
      reference
        "RFC 4253:
          The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish128-cbc {
      base encryption-alg-base;
      description
        "TWOFISH128-CBC";
      reference
        "RFC 4253:
          The Secure Shell (SSH) Transport Layer Protocol";
    }
```

```
     identity aes256-cbc {
       base encryption-alg-base;
       description
         "AES256-CBC";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity aes192-cbc {
       base encryption-alg-base;
       description
         "AES192-CBC";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity aes128-cbc {
       base encryption-alg-base;
       description
         "AES128-CBC";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity serpent256-cbc {
       base encryption-alg-base;
       description
         "SERPENT256-CBC";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity serpent192-cbc {
       base encryption-alg-base;
       description
         "SERPENT192-CBC";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity serpent128-cbc {
       base encryption-alg-base;
       description
```

```
        "SERPENT128-CBC";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity arcfour {
      base encryption-alg-base;
      status obsolete;
      description
        "ARCFOUR";
      reference
        "RFC 8758:
           Deprecating RC4 in Secure Shell (SSH)";
    }

    identity idea-cbc {
      base encryption-alg-base;
      description
        "IDEA-CBC";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity cast128-cbc {
      base encryption-alg-base;
      description
        "CAST128-CBC";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity none {
      base encryption-alg-base;
      description
        "NONE";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity des-cbc {
      base encryption-alg-base;
      status obsolete;
      description
        "DES-CBC";
```

```
      reference
        "FIPS 46-3:
           Data Encryption Standard (DES)";
    }

    identity arcfour128 {
      base encryption-alg-base;
      status obsolete;
      description
        "ARCFOUR128";
      reference
        "RFC 8758:
           Deprecating RC4 in Secure Shell (SSH)";
    }

    identity arcfour256 {
      base encryption-alg-base;
      status obsolete;
      description
        "ARCFOUR256";
      reference
        "RFC 8758:
           Deprecating RC4 in Secure Shell (SSH)";
    }

    identity aes128-ctr {
      base encryption-alg-base;
      description
        "AES128-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity aes192-ctr {
      base encryption-alg-base;
      description
        "AES192-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity aes256-ctr {
      base encryption-alg-base;
      description
        "AES256-CTR";
      reference
```

```
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity triple-des-ctr { // YANG IDs cannot begin with a number
      base encryption-alg-base;
      description
        "3DES-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity blowfish-ctr {
      base encryption-alg-base;
      description
        "BLOWFISH-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity twofish128-ctr {
      base encryption-alg-base;
      description
        "TWOFISH128-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity twofish192-ctr {
      base encryption-alg-base;
      description
        "TWOFISH192-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity twofish256-ctr {
      base encryption-alg-base;
      description
        "TWOFISH256-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }
```

```
    identity serpent128-ctr {
      base encryption-alg-base;
      description
        "SERPENT128-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity serpent192-ctr {
      base encryption-alg-base;
      description
        "SERPENT192-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity serpent256-ctr {
      base encryption-alg-base;
      description
        "SERPENT256-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity idea-ctr {
      base encryption-alg-base;
      description
        "IDEA-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity cast128-ctr {
      base encryption-alg-base;
      description
        "CAST128-CTR";
      reference
        "RFC 4344:
           The Secure Shell (SSH) Transport Layer Encryption Modes";
    }

    identity aead-aes-128-gcm {
      base encryption-alg-base;
      description
```

```
        "AEAD_AES_128_GCM";
      reference
        "RFC 5647:
           AES Galois Counter Mode for the
           Secure Shell Transport Layer Protocol";
    }

    identity aead-aes-256-gcm {
      base encryption-alg-base;
      description
        "AEAD_AES_256_GCM";
      reference
        "RFC 5647:
           AES Galois Counter Mode for the
           Secure Shell Transport Layer Protocol";
    }

    // Protocol-accessible Nodes

    container supported-algorithms {
      config false;
      description
        "A container for a list of encryption algorithms
         supported by the server.";
      leaf-list supported-algorithm {
        type encryption-algorithm-ref;
        description
          "A encryption algorithm supported by the server.";
      }
    }

  }

  <CODE ENDS>
```

A.2. Initial Module for the "MAC Algorithm Names" Registry

A.2.1. Data Model Overview

This section provides an overview of the "iana-ssh-mac-algs" module in terms of its identities and protocol-accessible nodes.

A.2.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-MAC-ALGS].

```
Identities:
  +-- mac-alg-base
    +-- <identity-name from IANA registry>

    | The diagram above uses syntax that is similar to but not
    | defined in [RFC8340].
```

A.2.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-mac-algs" module:

```
Typedefs:
  identityref
    +-- mac-algorithm-ref

    | The diagram above uses syntax that is similar to but not
    | defined in [RFC8340].
```

Comments:

* The typedef defined in the "iana-ssh-mac-algs" module extends the "identityref" type defined in [RFC7950].

A.2.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-mac-algs" module:

```
module: iana-ssh-mac-algs
  +--ro supported-algorithms
     +--ro supported-algorithm*   mac-algorithm-ref
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

A.2.2. Example Usage

The following example illustrates operational state data indicating the SSH MAC algorithms supported by the server:

```
   <supported-algorithms
     xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs"
     xmlns:sshma="urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs">
     <supported-algorithm>sshma:hmac-sha2-256</supported-algorithm>
     <supported-algorithm>sshma:hmac-sha2-512</supported-algorithm>
     <supported-algorithm>sshma:aead-aes-256-gcm</supported-algorithm>
   </supported-algorithms>
```

A.2.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
   <CODE BEGINS> file "iana-ssh-mac-algs@2021-06-01.yang"

   module iana-ssh-mac-algs {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs";
     prefix sshma;

     organization
       "Internet Assigned Numbers Authority (IANA)";

     contact
       "Postal: ICANN
                12025 Waterfront Drive, Suite 300
                Los Angeles, CA  90094-2536
                United States of America
        Tel:    +1 310 301 5800
        Email:  iana@iana.org";

     description
       "This module defines identities for the MAC algorithms
        defined in the 'MAC Algorithm Names' sub-registry of the
        'Secure Shell (SSH) Protocol Parameters' registry maintained
        by IANA.

        Copyright (c) 2021 IETF Trust and the persons identified as
        authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).
```

```
        The initial version of this YANG module is part of RFC EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
        itself for full legal notices.";

    revision 2021-06-01 {
      description
        "Initial version";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Typedefs

    typedef mac-algorithm-ref {
      type identityref {
        base "mac-alg-base";
      }
      description
        "A reference to a SSH mac algorithm identifier.";
    }


    // Identities

    identity mac-alg-base {
      description
        "Base identity used to identify message authentication
         code (MAC) algorithms.";
    }

    identity hmac-sha1 {
      base mac-alg-base;
      description
        "HMAC-SHA1";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity hmac-sha1-96 {
      base mac-alg-base;
      description
        "HMAC-SHA1-96";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }
```

```
     identity hmac-md5 {
       base mac-alg-base;
       description
         "HMAC-MD5";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity hmac-md5-96 {
       base mac-alg-base;
       description
         "HMAC-MD5-96";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity none {
       base mac-alg-base;
       description
         "NONE";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity aead-aes-128-gcm {
       base mac-alg-base;
       description
         "AEAD_AES_128_GCM";
       reference
         "RFC 5647:
            AES Galois Counter Mode for the
            Secure Shell Transport Layer Protocol";
     }

     identity aead-aes-256-gcm {
       base mac-alg-base;
       description
         "AEAD_AES_256_GCM";
       reference
         "RFC 5647:
            AES Galois Counter Mode for the
            Secure Shell Transport Layer Protocol";
     }

     identity hmac-sha2-256 {
```

```
      base mac-alg-base;
      description
        "HMAC-SHA2-256";
      reference
        "RFC 6668:
           SHA-2 Data Integrity Verification for the
           Secure Shell (SSH) Transport Layer Protocol";
    }

    identity hmac-sha2-512 {
      base mac-alg-base;
      description
        "HMAC-SHA2-512";
      reference
        "RFC 6668:
           SHA-2 Data Integrity Verification for the
           Secure Shell (SSH) Transport Layer Protocol";
    }

    // Protocol-accessible Nodes

    container supported-algorithms {
      config false;
      description
        "A container for a list of MAC algorithms
         supported by the server.";
      leaf-list supported-algorithm {
        type mac-algorithm-ref;
        description
          "A MAC algorithm supported by the server.";
      }
    }

  }

  <CODE ENDS>
```

A.3. Initial Module for the "Public Key Algorithm Names" Registry

A.3.1. Data Model Overview

This section provides an overview of the "iana-ssh-public-key-algs" module in terms of its identities and protocol-accessible nodes.

A.3.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-PUBKEY-ALGS].

```
Identities:
  +-- public-key-alg-base
    +-- <identity-name from IANA registry>

    |  The diagram above uses syntax that is similar to but not
    |  defined in [RFC8340].
```

A.3.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-public-key-algs" module:

```
Typedefs:
  identityref
    +-- public-key-algorithm-ref

    |  The diagram above uses syntax that is similar to but not
    |  defined in [RFC8340].
```

Comments:

* The typedef defined in the "iana-ssh-public-key-algs" module extends the "identityref" type defined in [RFC7950].

A.3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-public-key-algs" module:

```
module: iana-ssh-public-key-algs
  +--ro supported-algorithms
     +--ro supported-algorithm*   public-key-algorithm-ref
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

A.3.2. Example Usage

The following example illustrates operational state data indicating the SSH public key algorithms supported by the server:

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs"
  xmlns:sshpka="urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs\
">
  <supported-algorithm>sshpka:rsa-sha2-256</supported-algorithm>
  <supported-algorithm>sshpka:rsa-sha2-512</supported-algorithm>
  <supported-algorithm>sshpka:spki-sign-rsa</supported-algorithm>
  <supported-algorithm>sshpka:pgp-sign-dss</supported-algorithm>
  <supported-algorithm>sshpka:x509v3-rsa2048-sha256</supported-algor\
ithm>
  <supported-algorithm>sshpka:ecdsa-sha2-nistp256</supported-algorit\
hm>
  <supported-algorithm>sshpka:ecdsa-sha2-1.3.132.0.37</supported-alg\
orithm>
  <supported-algorithm>sshpka:ssh-ed25519</supported-algorithm>
</supported-algorithms>
```

A.3.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-ssh-public-key-algs@2021-06-01.yang"

module iana-ssh-public-key-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs";
  prefix sshpka;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
     Tel:    +1 310 301 5800
     Email:  iana@iana.org";
```

```
    description
      "This module defines identities for the public key algorithms
       defined in the 'Public Key Algorithm Names' sub-registry of the
       'Secure Shell (SSH) Protocol Parameters' registry maintained
       by IANA.

       Copyright (c) 2021 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

       Redistribution and use in source and binary forms, with
       or without modification, is permitted pursuant to, and
       subject to the license terms contained in, the Revised
       BSD License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       The initial version of this YANG module is part of RFC EEEE
       (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
       itself for full legal notices.";

    revision 2021-06-01 {
      description
        "Initial version";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Typedefs

    typedef public-key-algorithm-ref {
      type identityref {
        base "public-key-alg-base";
      }
      description
        "A reference to a SSH public key algorithm identifier.";
    }


    // Identities

    identity public-key-alg-base {
      description
        "Base identity used to identify public key algorithms.";
    }

    identity ssh-dss {
      base public-key-alg-base;
      description
```

```
        "SSH-DSS";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity ssh-rsa {
      base public-key-alg-base;
      description
        "SSH-RSA";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity rsa-sha2-256 {
      base public-key-alg-base;
      description
        "RSA-SHA2-256";
      reference
        "RFC 8332:
           Use of RSA Keys with SHA-256 and SHA-512
           in the Secure Shell (SSH) Protocol";
    }

    identity rsa-sha2-512 {
      base public-key-alg-base;
      description
        "RSA-SHA2-512";
      reference
        "RFC 8332:
           Use of RSA Keys with SHA-256 and SHA-512
           in the Secure Shell (SSH) Protocol";
    }

    identity spki-sign-rsa {
      base public-key-alg-base;
      description
        "SPKI-SIGN-RSA";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity spki-sign-dss {
      base public-key-alg-base;
      description
        "SPKI-SIGN-DSS";
```

```
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity pgp-sign-rsa {
      base public-key-alg-base;
      description
        "PGP-SIGN-RSA";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity pgp-sign-dss {
      base public-key-alg-base;
      description
        "PGP-SIGN-DSS";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity null {
      base public-key-alg-base;
      description
        "NULL";
      reference
        "RFC 4462:
           Generic Security Service Application Program Interface
           (GSS-API) Authentication and Key Exchange for the
           Secure Shell (SSH) Protocol";
    }

    identity ecdsa-sha2-nistp256 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-NISTP256 (secp256r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-nistp384 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-NISTP384 (secp384r1)";
```

```
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-nistp521 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-NISTP521 (secp521r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.3.132.0.1 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.2.840.10045.3.1.1 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.3.132.0.33 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.3.132.0.26 {
      base public-key-alg-base;
```

```
      description
        "ECDSA-SHA2-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.3.132.0.27 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.3.132.0.16 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.3.132.0.36 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-1.3.132.0.37 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }
```

```
    identity ecdsa-sha2-1.3.132.0.38 {
      base public-key-alg-base;
      description
        "ECDSA-SHA2-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity x509v3-ssh-dss {
      base public-key-alg-base;
      description
        "X509V3-SSH-DSS";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ssh-rsa {
      base public-key-alg-base;
      description
        "X509V3-SSH-RSA";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-rsa2048-sha256 {
      base public-key-alg-base;
      description
        "X509V3-RSA2048-SHA256";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp256 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-NISTP256 (secp256r1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp384 {
      base public-key-alg-base;
```

```
      description
        "X509V3-ECDSA-SHA2-NISTP384 (secp384r1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp521 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-NISTP521 (secp521r1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.1 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.2.840.10045.3.1.1 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.33 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.26 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.26 (nistk233, sect233k1)";
      reference
```

```
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.27 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.16 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.36 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.37 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-sha2-1.3.132.0.38 {
      base public-key-alg-base;
      description
        "X509V3-ECDSA-SHA2-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }
```

```
    identity ssh-ed25519 {
      base public-key-alg-base;
      description
        "SSH-ED25519";
      reference
        "RFC 8709:
           Ed25519 and Ed448 Public Key Algorithms for the
           Secure Shell (SSH) Protocol";
    }

    identity ssh-ed448 {
      base public-key-alg-base;
      description
        "SSH-ED448";
      reference
        "RFC 8709:
           Ed25519 and Ed448 Public Key Algorithms for the
           Secure Shell (SSH) Protocol";
    }

    // Protocol-accessible Nodes

    container supported-algorithms {
      config false;
      description
        "A container for a list of public key algorithms
         supported by the server.";
      leaf-list supported-algorithm {
        type public-key-algorithm-ref;
        description
          "A public key algorithm supported by the server.";
      }
    }

  }

  <CODE ENDS>
```

A.4. Initial Module for the "Key Exchange Method Names" Registry

A.4.1. Data Model Overview

This section provides an overview of the "iana-ssh-key-exchange-algs" module in terms of its identities and protocol-accessible nodes.

A.4.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-KEYEX-ALGS].

```
Identities:
  +-- key-exchange-alg-base
    +-- <identity-name from IANA registry>

    |  The diagram above uses syntax that is similar to but not
    |  defined in [RFC8340].
```

A.4.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-key-exchange-algs" module:

```
Typedefs:
  identityref
    +-- key-exchange-algorithm-ref

    |  The diagram above uses syntax that is similar to but not
    |  defined in [RFC8340].
```

Comments:

* The typedef defined in the "iana-ssh-key-exchange-algs" module extends the "identityref" type defined in [RFC7950].

A.4.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-key-exchange-algs" module:

```
module: iana-ssh-key-exchange-algs
  +--ro supported-algorithms
     +--ro supported-algorithm*   key-exchange-algorithm-ref
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

A.4.2. Example Usage

The following example illustrates operational state data indicating the SSH key exchange algorithms supported by the server:

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs"
  xmlns:sshkea="urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-al\
gs">
  <supported-algorithm>sshkea:diffie-hellman-group-exchange-sha256</\
supported-algorithm>
  <supported-algorithm>sshkea:ecdh-sha2-nistp256</supported-algorith\
m>
  <supported-algorithm>sshkea:rsa2048-sha256</supported-algorithm>
  <supported-algorithm>sshkea:gss-group1-sha1-curve25519-sha256</sup\
ported-algorithm>
  <supported-algorithm>sshkea:gss-group14-sha1-nistp256</supported-a\
lgorithm>
  <supported-algorithm>sshkea:gss-gex-sha1-nistp256</supported-algor\
ithm>
  <supported-algorithm>sshkea:gss-group14-sha256-1.2.840.10045.3.1.1\
</supported-algorithm>
  <supported-algorithm>sshkea:curve25519-sha256</supported-algorithm>
</supported-algorithms>
```

A.4.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-ssh-key-exchange-algs@2021-06-01.yang"

module iana-ssh-key-exchange-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs";
  prefix sshkea;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
```

```
        Tel:    +1 310 301 5800
        Email:  iana@iana.org";

    description
      "This module defines identities for the key exchange algorithms
       defined in the 'Key Exchange Method Names' sub-registry of the
       'Secure Shell (SSH) Protocol Parameters' registry maintained
       by IANA.

       Copyright (c) 2021 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

       Redistribution and use in source and binary forms, with
       or without modification, is permitted pursuant to, and
       subject to the license terms contained in, the Revised
       BSD License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       The initial version of this YANG module is part of RFC EEEE
       (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
       itself for full legal notices.";

    revision 2021-06-01 {
      description
        "Initial version";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Typedefs

    typedef key-exchange-algorithm-ref {
      type identityref {
        base "key-exchange-alg-base";
      }
      description
        "A reference to a SSH key exchange algorithm identifier.";
    }


    // Identities

    identity key-exchange-alg-base {
      description
        "Base identity used to identify key exchange algorithms.";
    }
```

```
  identity diffie-hellman-group-exchange-sha1 {
    base key-exchange-alg-base;
    description
      "DIFFIE-HELLMAN-GROUP-EXCHANGE-SHA1";
    reference
      "RFC 4419:
         Diffie-Hellman Group Exchange for the
         Secure Shell (SSH) Transport Layer Protocol";
  }

  identity diffie-hellman-group-exchange-sha256 {
    base key-exchange-alg-base;
    description
      "DIFFIE-HELLMAN-GROUP-EXCHANGE-SHA256";
    reference
      "RFC 4419:
         Diffie-Hellman Group Exchange for the
         Secure Shell (SSH) Transport Layer Protocol";
  }

  identity diffie-hellman-group1-sha1 {
    base key-exchange-alg-base;
    description
      "DIFFIE-HELLMAN-GROUP1-SHA1";
    reference
      "RFC 4253:
         The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity diffie-hellman-group14-sha1 {
    base key-exchange-alg-base;
    description
      "DIFFIE-HELLMAN-GROUP14-SHA1";
    reference
      "RFC 4253:
         The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity diffie-hellman-group14-sha256 {
    base key-exchange-alg-base;
    description
      "DIFFIE-HELLMAN-GROUP14-SHA256";
    reference
      "RFC 8268:
         More Modular Exponentiation (MODP) Diffie-Hellman (DH)
         Key Exchange (KEX) Groups for Secure Shell (SSH)";
  }
```

```
    identity diffie-hellman-group15-sha512 {
      base key-exchange-alg-base;
      description
        "DIFFIE-HELLMAN-GROUP15-SHA512";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity diffie-hellman-group16-sha512 {
      base key-exchange-alg-base;
      description
        "DIFFIE-HELLMAN-GROUP16-SHA512";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity diffie-hellman-group17-sha512 {
      base key-exchange-alg-base;
      description
        "DIFFIE-HELLMAN-GROUP17-SHA512";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity diffie-hellman-group18-sha512 {
      base key-exchange-alg-base;
      description
        "DIFFIE-HELLMAN-GROUP18-SHA512";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity ecdh-sha2-nistp256 {
      base key-exchange-alg-base;
      description
        "ECDH-SHA2-NISTP256 (secp256r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
```

```
    }

    identity ecdh-sha2-nistp384 {
      base key-exchange-alg-base;
      description
        "ECDH-SHA2-NISTP384 (secp384r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdh-sha2-nistp521 {
      base key-exchange-alg-base;
      description
        "ECDH-SHA2-NISTP521 (secp521r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "ECDH-SHA2-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
        "ECDH-SHA2-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.33 {
      base key-exchange-alg-base;
      description
        "ECDH-SHA2-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 5656:
```

```
            Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
      }

      identity ecdh-sha2-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
          "ECDH-SHA2-1.3.132.0.26 (nistk233, sect233k1)";
        reference
          "RFC 5656:
             Elliptic Curve Algorithm Integration in the
             Secure Shell Transport Layer";
      }

      identity ecdh-sha2-1.3.132.0.27 {
        base key-exchange-alg-base;
        description
          "ECDH-SHA2-1.3.132.0.27 (nistb233, sect233r1)";
        reference
          "RFC 5656:
             Elliptic Curve Algorithm Integration in the
             Secure Shell Transport Layer";
      }

      identity ecdh-sha2-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
          "ECDH-SHA2-1.3.132.0.16 (nistk283, sect283k1)";
        reference
          "RFC 5656:
             Elliptic Curve Algorithm Integration in the
             Secure Shell Transport Layer";
      }

      identity ecdh-sha2-1.3.132.0.36 {
        base key-exchange-alg-base;
        description
          "ECDH-SHA2-1.3.132.0.36 (nistk409, sect409k1)";
        reference
          "RFC 5656:
             Elliptic Curve Algorithm Integration in the
             Secure Shell Transport Layer";
      }

      identity ecdh-sha2-1.3.132.0.37 {
        base key-exchange-alg-base;
        description
          "ECDH-SHA2-1.3.132.0.37 (nistb409, sect409r1)";
```

```
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "ECDH-SHA2-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity ecmqv-sha2 {
      base key-exchange-alg-base;
      description
        "ECMQV-SHA2";
      reference
        "RFC 5656:
           Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer";
    }

    identity gss-group1-sha1-nistp256 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-nistp384 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-group1-sha1-nistp521 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.1 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.33 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.26 {
      base key-exchange-alg-base;
      status deprecated;
      description
```

```
        "GSS-GROUP1-SHA1-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.27 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.16 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.36 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.37 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
```

```
          (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.38 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-curve25519-sha256 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-curve448-sha512 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP1-SHA1-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-nistp256 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-nistp384 {
```

```
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-nistp521 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-1.3.132.0.1 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-1.3.132.0.33 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-1.3.132.0.33 (nistp224, secp224r1)";
```

```
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-1.3.132.0.26 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-1.3.132.0.27 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-1.3.132.0.16 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-1.3.132.0.36 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GROUP14-SHA1-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
```

```
  }

  identity gss-group14-sha1-1.3.132.0.37 {
    base key-exchange-alg-base;
    status deprecated;
    description
      "GSS-GROUP14-SHA1-1.3.132.0.37 (nistb409, sect409r1)";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group14-sha1-1.3.132.0.38 {
    base key-exchange-alg-base;
    status deprecated;
    description
      "GSS-GROUP14-SHA1-1.3.132.0.38 (nistt571, sect571k1)";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group14-sha1-curve25519-sha256 {
    base key-exchange-alg-base;
    status deprecated;
    description
      "GSS-GROUP14-SHA1-CURVE25519-SHA256";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group14-sha1-curve448-sha512 {
    base key-exchange-alg-base;
    status deprecated;
    description
      "GSS-GROUP14-SHA1-CURVE448-SHA512";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-gex-sha1-nistp256 {
    base key-exchange-alg-base;
```

```
      status deprecated;
      description
        "GSS-GEX-SHA1-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-nistp384 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-nistp521 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.3.132.0.1 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
```

```
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.3.132.0.33 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.3.132.0.26 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.3.132.0.27 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.3.132.0.16 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-gex-sha1-1.3.132.0.36 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.3.132.0.37 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-1.3.132.0.38 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-curve25519-sha256 {
      base key-exchange-alg-base;
      status deprecated;
      description
        "GSS-GEX-SHA1-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-gex-sha1-curve448-sha512 {
      base key-exchange-alg-base;
      status deprecated;
      description
```

```
        "GSS-GEX-SHA1-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity rsa1024-sha1 {
      base key-exchange-alg-base;
      description
        "RSA1024-SHA1";
      reference
        "RFC 4432:
           RSA Key Exchange for the Secure Shell (SSH)
           Transport Layer Protocol";
    }

    identity rsa2048-sha256 {
      base key-exchange-alg-base;
      description
        "RSA2048-SHA256";
      reference
        "RFC 4432:
           RSA Key Exchange for the Secure Shell (SSH)
           Transport Layer Protocol";
    }

    identity ext-info-s {
      base key-exchange-alg-base;
      description
        "EXT-INFO-S";
      reference
        "RFC 8308:
           Extension Negotiation in the Secure Shell (SSH) Protocol";
    }

    identity ext-info-c {
      base key-exchange-alg-base;
      description
        "EXT-INFO-C";
      reference
        "RFC 8308:
           Extension Negotiation in the Secure Shell (SSH) Protocol";
    }

    identity gss-group14-sha256-nistp256 {
      base key-exchange-alg-base;
      description
```

```
        "GSS-GROUP14-SHA256-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-nistp384 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-nistp521 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.3.132.0.33 {
```

```
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.3.132.0.26 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.3.132.0.27 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.3.132.0.16 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-group14-sha256-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha256-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP14-SHA256-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP15-SHA512-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
```

```
  }

  identity gss-group15-sha512-nistp384 {
    base key-exchange-alg-base;
    description
      "GSS-GROUP15-SHA512-NISTP384 (secp384r1)";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group15-sha512-nistp521 {
    base key-exchange-alg-base;
    description
      "GSS-GROUP15-SHA512-NISTP521 (secp521r1)";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group15-sha512-1.3.132.0.1 {
    base key-exchange-alg-base;
    description
      "GSS-GROUP15-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group15-sha512-1.2.840.10045.3.1.1 {
    base key-exchange-alg-base;
    description
      "GSS-GROUP15-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
    reference
      "RFC 8732:
         Generic Security Service Application Program Interface
         (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group15-sha512-1.3.132.0.33 {
    base key-exchange-alg-base;
    description
      "GSS-GROUP15-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
    reference
      "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group15-sha512-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP15-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group15-sha512-1.3.132.0.27 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP15-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group15-sha512-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP15-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group15-sha512-1.3.132.0.36 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP15-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group15-sha512-1.3.132.0.37 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP15-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
```

```
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP15-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP15-SHA512-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP15-SHA512-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-nistp384 {
      base key-exchange-alg-base;
```

```
      description
        "GSS-GROUP16-SHA512-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-nistp521 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-1.3.132.0.33 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-group16-sha512-1.3.132.0.26 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-1.3.132.0.27 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-1.3.132.0.16 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
```

```
    }

    identity gss-group16-sha512-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group16-sha512-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP16-SHA512-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-nistp384 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group17-sha512-nistp521 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP17-SHA512-NISTP521 (secp521r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group17-sha512-1.3.132.0.1 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP17-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group17-sha512-1.2.840.10045.3.1.1 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP17-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group17-sha512-1.3.132.0.33 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP17-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group17-sha512-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP17-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
```

```
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.27 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.16 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.38 {
      base key-exchange-alg-base;
```

```
      description
        "GSS-GROUP17-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP17-SHA512-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-nistp384 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-group18-sha512-nistp521 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.3.132.0.33 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.3.132.0.26 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
```

```
    }

    identity gss-group18-sha512-1.3.132.0.27 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.3.132.0.16 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-GROUP18-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group18-sha512-curve25519-sha256 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP18-SHA512-CURVE25519-SHA256";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-group18-sha512-curve448-sha512 {
        base key-exchange-alg-base;
        description
          "GSS-GROUP18-SHA512-CURVE448-SHA512";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp256-sha256-nistp256 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP256-SHA256-NISTP256 (secp256r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp256-sha256-nistp384 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP256-SHA256-NISTP384 (secp384r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp256-sha256-nistp521 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP256-SHA256-NISTP521 (secp521r1)";
```

```
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
       "GSS-NISTP256-SHA256-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.33 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.26 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.27 {
      base key-exchange-alg-base;
```

```
      description
        "GSS-NISTP256-SHA256-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.16 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-nistp256-sha256-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP256-SHA256-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-nistp384 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-nistp521 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
```

```
    }

    identity gss-nistp384-sha384-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
       "GSS-NISTP384-SHA384-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.3.132.0.33 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.3.132.0.26 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.3.132.0.27 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp384-sha384-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP384-SHA384-1.3.132.0.16 (nistk283, sect283k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp384-sha384-1.3.132.0.36 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP384-SHA384-1.3.132.0.36 (nistk409, sect409k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp384-sha384-1.3.132.0.37 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP384-SHA384-1.3.132.0.37 (nistb409, sect409r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp384-sha384-1.3.132.0.38 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP384-SHA384-1.3.132.0.38 (nistt571, sect571k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-nistp384-sha384-curve25519-sha256 {
        base key-exchange-alg-base;
        description
          "GSS-NISTP384-SHA384-CURVE25519-SHA256";
```

```
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP384-SHA384-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-nistp384 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-nistp521 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.3.132.0.1 {
      base key-exchange-alg-base;
```

```
      description
        "GSS-NISTP521-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
       "GSS-NISTP521-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.3.132.0.33 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.3.132.0.26 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.3.132.0.27 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-nistp521-sha512-1.3.132.0.16 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp521-sha512-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
```

```
    }

    identity gss-nistp521-sha512-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-NISTP521-SHA512-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-nistp384 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-nistp521 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-curve25519-sha256-1.2.840.10045.3.1.1 {
        base key-exchange-alg-base;
        description
          "GSS-CURVE25519-SHA256-1.2.840.10045.3.1.1 (nistp192,
           secp192r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-curve25519-sha256-1.3.132.0.33 {
        base key-exchange-alg-base;
        description
          "GSS-CURVE25519-SHA256-1.3.132.0.33 (nistp224, secp224r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-curve25519-sha256-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
          "GSS-CURVE25519-SHA256-1.3.132.0.26 (nistk233, sect233k1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-curve25519-sha256-1.3.132.0.27 {
        base key-exchange-alg-base;
        description
          "GSS-CURVE25519-SHA256-1.3.132.0.27 (nistb233, sect233r1)";
        reference
          "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
      }

      identity gss-curve25519-sha256-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
```

```
        "GSS-CURVE25519-SHA256-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-curve448-sha512 {
```

```
      base key-exchange-alg-base;
      description
        "GSS-CURVE25519-SHA256-CURVE448-SHA512";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-nistp256 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-NISTP256 (secp256r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-nistp384 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-NISTP384 (secp384r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-nistp521 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-NISTP521 (secp521r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-1.3.132.0.1 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }
```

```
    identity gss-curve448-sha512-1.2.840.10045.3.1.1 {
      base key-exchange-alg-base;
      description
       "GSS-CURVE448-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-1.3.132.0.33 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-1.3.132.0.26 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-1.3.132.0.27 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-1.3.132.0.16 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
```

```
    }

    identity gss-curve448-sha512-1.3.132.0.36 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-1.3.132.0.37 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-1.3.132.0.38 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-curve25519-sha256 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-CURVE25519-SHA256";
      reference
        "RFC 8732:
           Generic Security Service Application Program Interface
           (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve448-sha512-curve448-sha512 {
      base key-exchange-alg-base;
      description
        "GSS-CURVE448-SHA512-CURVE448-SHA512";
      reference
        "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
      }

      identity curve25519-sha256 {
        base key-exchange-alg-base;
        description
          "CURVE25519-SHA256";
        reference
          "RFC 8731:
             Secure Shell (SSH) Key Exchange Method
             Using Curve25519 and Curve448";
      }

      identity curve448-sha512 {
        base key-exchange-alg-base;
        description
          "CURVE448-SHA512";
        reference
          "RFC 8731:
             Secure Shell (SSH) Key Exchange Method
             Using Curve25519 and Curve448";
      }

      // Protocol-accessible Nodes

      container supported-algorithms {
        config false;
        description
          "A container for a list of key exchange algorithms
           supported by the server.";
        leaf-list supported-algorithm {
          type key-exchange-algorithm-ref;
          description
            "A key exchange algorithm supported by the server.";
        }
      }

    }

    <CODE ENDS>
```

Appendix B. Change Log

This section is to be removed before publishing as an RFC.

B.1. 00 to 01

* Noted that '0.0.0.0' and '::' might have special meanings.

* Renamed "keychain" to "keystore".

B.2. 01 to 02

* Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.

* Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.

* Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

B.3. 02 to 03

* Removed 'RESTRICTED' enum from 'password' leaf type.

* Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

* Fixed description statement for leaf 'trusted-ca-certs'.

B.4. 03 to 04

* Change title to "YANG Groupings for SSH Clients and SSH Servers"

* Added reference to RFC 6668

* Added RFC 8174 to Requirements Language Section.

* Enhanced description statement for ietf-ssh-server's "trusted-ca-certs" leaf.

* Added mandatory true to ietf-ssh-client's "client-auth" 'choice' statement.

* Changed the YANG prefix for module ietf-ssh-common from 'sshcom' to 'sshcmn'.

* Removed the compression algorithms as they are not commonly configurable in vendors' implementations.

* Updating descriptions in transport-params-grouping and the servers's usage of it.

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated YANG to use typedefs around leafrefs to common keystore paths

* Now inlines key and certificates (no longer a leafref to keystore)

B.5. 04 to 05

* Merged changes from co-author.

B.6. 05 to 06

* Updated to use trust anchors from trust-anchors draft (was keystore draft)

* Now uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

B.7. 06 to 07

* factored the ssh-[client|server]-groupings into more reusable groupings.

* added if-feature statements for the new "ssh-host-keys" and "x509-certificates" features defined in draft-ietf-netconf-trust-anchors.

B.8. 07 to 08

* Added a number of compatibility matrices to Section 5 (thanks Frank!)

* Clarified that any configured "host-key-alg" values need to be compatible with the configured private key.

B.9. 08 to 09

* Updated examples to reflect update to groupings defined in the keystore -09 draft.

* Add SSH keepalives features and groupings.

* Prefixed top-level SSH grouping nodes with 'ssh-' and support mashups.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

* Reformatted the YANG modules.

B.11. 10 to 11

* Reformatted lines causing folding to occur.

B.12. 11 to 12

* Collapsed all the inner groupings into the top-level grouping.

* Added a top-level "demux container" inside the top-level grouping.

* Added NACM statements and updated the Security Considerations section.

* Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

B.13. 12 to 13

* Removed the "demux containers", floating the nacm:default-deny-write to each descendant node, and adding a note to model designers regarding the potential need to add their own demux containers.

* Fixed a couple references (section 2 --> section 3)

* In the server model, replaced <client-cert-auth> with <client-authentication> and introduced 'local-or-external' choice.

B.14. 13 to 14

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

B.15. 14 to 15

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:host-keys-ref" or "ts:certificates-ref" to a container that uses "ts:local-or-truststore-host-keys-grouping" or "ts:local-or-truststore-certs-grouping".

B.16. 15 to 16

* Removed unnecessary if-feature statements in the -client and -server modules.

* Cleaned up some description statements in the -client and -server modules.

* Fixed a canonical ordering issue in ietf-ssh-common detected by new pyang.

B.17. 16 to 17

* Removed choice local-or-external by removing the 'external' case and flattening the 'local' case and adding a "local-users-supported" feature.

* Updated examples to include the "*-key-format" nodes.

* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:ssh-public-key-format" (must expr for ref'ed keys are TBD).

B.18. 17 to 18

* Removed leaf-list 'other' from ietf-ssh-server.

* Removed unused 'external-client-auth-supported' feature.

* Added features client-auth-password, client-auth-hostbased, and client-auth-none.

* Renamed 'host-key' to 'public-key' for when refering to 'publickey' based auth.

* Added new feature-protected 'hostbased' and 'none' to the 'user' node's config.

* Added new feature-protected 'hostbased' and 'none' to the 'client-identity' node's config.

* Updated examples to reflect new "bag" addition to truststore.

* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.

* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags.

* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

B.19. 18 to 19

* Updated the "keepalives" containers to address Michal Vasko's request to align with RFC 8071.

* Removed algorithm-mapping tables from the "SSH Common Model" section

* Removed 'algorithm' node from examples.

* Added feature "userauth-publickey"

* Removed "choice auth-type", as auth-types are not exclusive.

* Renamed both "client-certs" and "server-certs" to "ee-certs"

* Switch "must" to assert the public-key-format is "subject-public-key-info-format" when certificates are used.

* Added a "Note to Reviewers" note to first page.

B.20. 19 to 20

* Added a "must 'public-key or password or hostbased or none or certificate'" statement to the "user" node in ietf-ssh-client

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Moved the "ietf-ssh-common" module section to proceed the other two module sections.

* Updated the Security Considerations section.

B.21. 20 to 21

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

B.22. 21 to 22

* Cleaned up the SSH-client examples (i.e., removing FIXMEs)
* Fixed issues found by the SecDir review of the "keystore" draft.
* Updated the "ietf-ssh-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

B.23. 22 to 23

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

B.24. 23 to 24

* Removed the 'supported-authentication-methods' from {grouping ssh-server-grouping}/client-authentication.
* Added XML-comment above examples explaining the reason for the unexepected top-most element's presence.
* Added RFC-references to various 'feature' statements.
* Renamed "credentials" to "authentication methods"
* Renamed "client-auth-*" to "userauth-*"
* Renamed "client-identity-*" to "userauth-*"
* Fixed nits found by YANG Doctor reviews.
* Aligned modules with `pyang -f` formatting.
* Added a 'Contributors' section.

B.25. 24 to 25

* Moved algorithms in ietf-ssh-common (plus more) to IANA-maintained modules
* Added "config false" lists for algorithms supported by the server.
* Renamed "{ietf-ssh-client}userauth-*" to "client-ident-*"
* Renamed "{ietf-ssh-server}userauth-*" to "local-user-auth-*"
* Fixed issues found during YANG Doctor review.

* Fixed issues found during Secdir review.

B.26. 25 to 26

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

B.27. 26 to 27

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)

* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

* Created identityref-based typedefs for each of the four IANA alg identity bases.

* Added ietf-ssh-common:generate-public-key() RPC for discussion.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balazs Kovacs, Barry Leiba, Benoit Claise, Bert Wijnen, David Lamparter, Gary Wu, Juergen Schoenwaelder, Ladislav Lhotka, Liang Xia, Martin Bjoerklund, Mehmet Ersue, Michal Vasko, Phil Shafer, Radek Krejci, Sean Turner, Tom Petch.

Contributors

Special acknowledgement goes to Gary Wu for his work on the "ietf-ssh-common" module.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks
M. Scharf
Hochschule Esslingen

7 March 2022

YANG Groupings for TCP Clients and TCP Servers
draft-ietf-netconf-tcp-client-server-12

Abstract

This document defines three YANG 1.1 modules to support the configuration of TCP clients and TCP servers. The modules include basic parameters of a TCP connection relevant for client or server applications, as well as client configuration required for traversing proxies. The modules can be used either standalone or in conjunction with configuration of other stack protocol layers.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types

* DDDD --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 3
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
  1.4. Conventions 5
2. The "ietf-tcp-common" Module 6
  2.1. Data Model Overview 6
  2.2. Example Usage 8
  2.3. YANG Module 8
3. The "ietf-tcp-client" Module 11
  3.1. Data Model Overview 11
  3.2. Example Usage 13
  3.3. YANG Module 14
4. The "ietf-tcp-server" Module 21
  4.1. Data Model Overview 21
  4.2. Example Usage 23
  4.3. YANG Module 23
5. Security Considerations 26
  5.1. The "ietf-tcp-common" YANG Module 26
  5.2. The "ietf-tcp-client" YANG Module 26

5.3. The "ietf-tcp-server" YANG Module 27
6. IANA Considerations 28
6.1. The "IETF XML" Registry 28
6.2. The "YANG Module Names" Registry 28
7. References 29
7.1. Normative References 29
7.2. Informative References 29
Appendix A. Change Log 31
A.1. 00 to 01 31
A.2. 01 to 02 32
A.3. 02 to 03 32
A.4. 03 to 04 32
A.5. 04 to 05 32
A.6. 05 to 06 32
A.7. 06 to 07 32
A.8. 07 to 08 32
A.9. 08 to 09 33
A.10. 09 to 10 33
A.11. 10 to 11 33
A.12. 11 to 12 33
Acknowledgements 33
Authors' Addresses 33

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers (TCP is defined in [RFC0793]), either as standalone or in conjunction with configuration of other stack protocol layers.

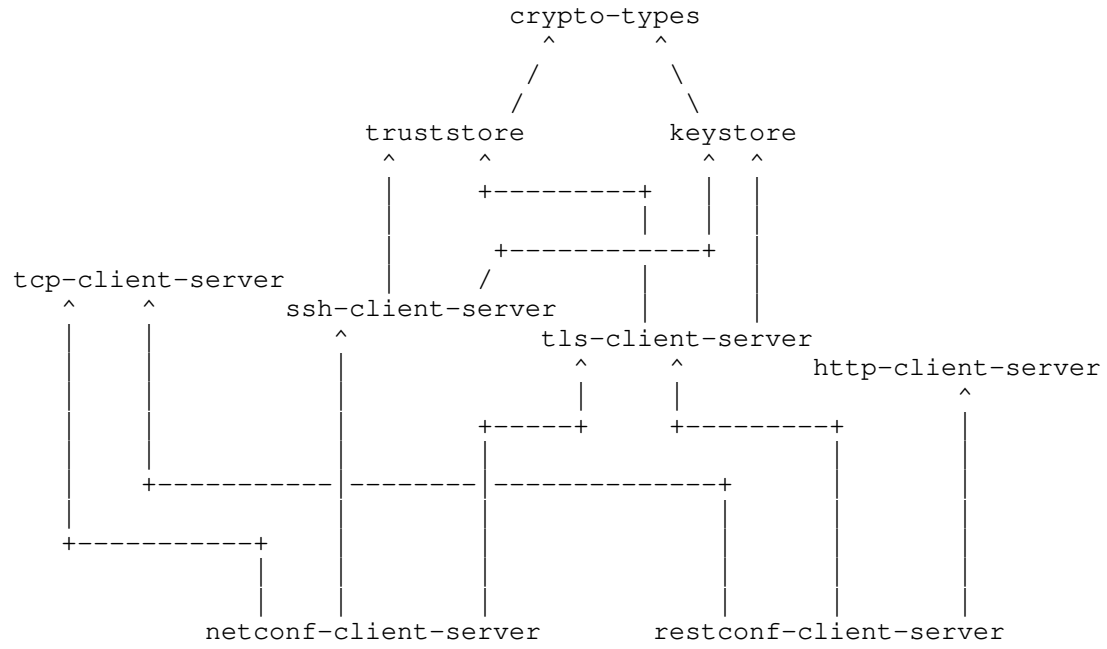
The modules focus on three different types of base TCP parameters that matter for TCP-based applications: First, the modules cover fundamental configuration of a TCP client or TCP server application, such as addresses and port numbers. Second, a reusable grouping enables modification of application-specific parameters for a TCP connections, such as use of TCP keep-alives. And third, client configuration for traversing proxies is included as well. In each case, the modules have a very narrow scope and focus on a minimum set of required parameters.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-tcp-common" Module

This section defines a YANG 1.1 module called "ietf-tcp-common". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-tcp-common" module in terms of its features and groupings.

2.1.1. Model Scope

This document defines a common "grouping" statement for basic TCP connection parameters that matter to applications. In some TCP stacks, such parameters can also directly be set by an application using system calls, such as the sockets API. The base YANG model in this document focuses on modeling TCP keep-alives. This base model can be extended as needed.

2.1.2. Features

The following diagram lists all the "feature" statements defined in the "ietf-tcp-common" module:

```
Features:
  +-- keepalives-supported
```

```
   |  The diagram above uses syntax that is similar to but not
   |  defined in [RFC8340].
```

2.1.3. Groupings

The "ietf-tcp-common" module defines the following "grouping" statement:

* tcp-common-grouping

This grouping is presented in the following subsection.

2.1.3.1. The "tcp-common-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tcp-common-grouping" grouping:

```
  grouping tcp-common-grouping
    +-- keepalives! {keepalives-supported}?
       +-- idle-time         uint16
       +-- max-probes        uint16
       +-- probe-interval    uint16
```

Comments:

* The "keepalives" node is a "presence" node so that the mandatory descendant nodes do not imply that keepalives must be configured.

* The "idle-time", "max-probes", and "probe-interval" nodes have the common meanings. Please see the YANG module in Section 2.3 for details.

2.1.4. Protocol-accessible Nodes

The "ietf-tcp-common" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

2.1.5. Guidelines for Configuring TCP Keep-Alives

Network stacks may include "keep-alives" in their TCP implementations, although this practice is not universally accepted. If keep-alives are included, [RFC1122] mandates that the application MUST be able to turn them on or off for each TCP connection, and that they MUST default to off.

Keep-alive mechanisms exist in many protocols. Depending on the protocol stack, TCP keep-alives may only be one out of several alternatives. Which mechanism(s) to use depends on the use case and application requirements. If keep-alives are needed by an application, it is RECOMMENDED that the aliveness check happens only at the protocol layers that are meaningful to the application.

A TCP keep-alive mechanism SHOULD only be invoked in server applications that might otherwise hang indefinitely and consume resources unnecessarily if a client crashes or aborts a connection during a network failure [RFC1122]. TCP keep-alives may consume significant resources both in the network and in endpoints (e.g., battery power). In addition, frequent keep-alives risk network congestion. The higher the frequency of keep-alives, the higher the overhead.

Given the cost of keep-alives, parameters have to be configured carefully:

* The default idle interval (leaf "idle-time") MUST default to no less than two hours, i.e., 7200 seconds [RFC1122]. A lower value MAY be configured, but keep-alive messages SHOULD NOT be transmitted more frequently than once every 15 seconds. Longer intervals SHOULD be used when possible.

* The maximum number of sequential keep-alive probes that can fail (leaf "max-probes") trades off responsiveness and robustness against packet loss. ACK segments that contain no data are not reliably transmitted by TCP. Consequently, if a keep-alive mechanism is implemented it MUST NOT interpret failure to respond to any specific probe as a dead connection [RFC1122]. Typically, a single-digit number should suffice.

* TCP implementations may include a parameter for the number of seconds between TCP keep-alive probes (leaf "probe-interval"). In order to avoid congestion, the time interval between probes MUST NOT be smaller than one second. Significantly longer intervals SHOULD be used. It is important to note that keep-alive probes (or replies) can get dropped due to network congestion. Sending further probe messages into a congested path after a short interval, without backing off timers, could cause harm and result in a congestion collapse. Therefore it is essential to pick a large, conservative value for this interval.

2.2. Example Usage

This section presents an example showing the "tcp-common-grouping" populated with some data.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-common xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-common">
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-common>
```

2.3. YANG Module

The ietf-tcp-common YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-common@2022-03-07.yang"

```
   module ietf-tcp-common {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-common";
     prefix tcpcmn;

     organization
       "IETF NETCONF (Network Configuration) Working Group and the
        IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

     contact
       "WG Web:   https://datatracker.ietf.org/wg/netconf
                  https://datatracker.ietf.org/wg/tcpm
        WG List:  NETCONF WG list <mailto:netconf@ietf.org>
                  TCPM WG list <mailto:tcpm@ietf.org>
        Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                  Michael Scharf
                  <mailto:michael.scharf@hs-esslingen.de>";

     description
       "This module defines reusable groupings for TCP commons that
        can be used as a basis for specific TCP common instances.

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC DDDD
        (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

     revision 2022-03-07 {
       description
         "Initial version";
       reference
         "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
```

```
  }

  // Features

  feature keepalives-supported {
    description
      "Indicates that keepalives are supported.";
  }

  // Groupings

  grouping tcp-common-grouping {
    description
      "A reusable grouping for configuring TCP parameters common
       to TCP connections as well as the operating system as a
       whole.";
    container keepalives {
      if-feature "keepalives-supported";
      presence
        "Indicates that keepalives are enabled.  This statement is
         present so the mandatory descendant nodes do not imply that
         this node must be configured.";
      description
        "Configures the keep-alive policy, to proactively test the
         aliveness of the TCP peer.  An unresponsive TCP peer is
         dropped after approximately (idle-time + max-probes
         * probe-interval) seconds.";
      leaf idle-time {
        type uint16 {
          range "1..max";
        }
        units "seconds";
        mandatory true;
        description
          "Sets the amount of time after which if no data has been
           received from the TCP peer, a TCP-level probe message
           will be sent to test the aliveness of the TCP peer.
           Two hours (7200 seconds) is safe value, per RFC 1122.";
        reference
          "RFC 1122:
            Requirements for Internet Hosts -- Communication Layers";
      }
      leaf max-probes {
        type uint16 {
          range "1..max";
        }
        mandatory true;
        description
```

```
            "Sets the maximum number of sequential keep-alive probes
             that can fail to obtain a response from the TCP peer
             before assuming the TCP peer is no longer alive.";
        }
        leaf probe-interval {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          mandatory true;
          description
            "Sets the time interval between failed probes. The interval
             SHOULD be significantly longer than one second in order to
             avoid harm on a congested link.";
        }
      } // container keepalives
    } // grouping tcp-common-grouping

}
```

<CODE ENDS>

3. The "ietf-tcp-client" Module

This section defines a YANG 1.1 module called "ietf-tcp-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-tcp-client" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tcp-client" module:

```
Features:
  +-- local-binding-supported
  +-- tcp-client-keepalives
  +-- proxy-connect
  +-- socks5-gss-api
  +-- socks5-username-password
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

3.1.2. Groupings

The "ietf-tcp-client" module defines the following "grouping" statement:

* tcp-client-grouping

This grouping is presented in the following subsection.

3.1.2.1. The "tcp-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tcp-client-grouping" grouping:

```
  grouping tcp-client-grouping
    +-- remote-address                inet:host
    +-- remote-port?                  inet:port-number
    +-- local-address?                inet:ip-address
    |       {local-binding-supported}?
    +-- local-port?                   inet:port-number
    |       {local-binding-supported}?
    +-- proxy-server! {proxy-connect}?
    |  +-- (proxy-type)
    |     +--:(socks4)
    |     |  +-- socks4-parameters
    |     |     +-- remote-address    inet:ip-address
    |     |     +-- remote-port?      inet:port-number
    |     +--:(socks4a)
    |     |  +-- socks4a-parameters
    |     |     +-- remote-address    inet:host
    |     |     +-- remote-port?      inet:port-number
    |     +--:(socks5)
    |        +-- socks5-parameters
    |           +-- remote-address               inet:host
    |           +-- remote-port?                 inet:port-number
    |           +-- authentication-parameters!
    |              +-- (auth-type)
    |                 +--:(gss-api) {socks5-gss-api}?
    |                 |  +-- gss-api
    |                 +--:(username-password)
    |                          {socks5-username-password}?
    |                    +-- username-password
    |                       +-- username                string
    |                       +---u ct:password-grouping
    +---u tcpcmn:tcp-common-grouping
```

Comments:

* The "remote-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, a hostname.

* The "remote-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* The "local-address" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), is not mandatory. Its default value is '0', indicating that the operating system can pick an arbitrary port number.

* The "proxy-server" node is enabled by a "feature" statement and, for servers that enable it, is a "presence" container so that the descendant "mandatory true" choice node does not imply that the proxy-server node must be configured.

* This grouping uses the "tcp-common-grouping" grouping discussed in Section 2.1.3.1.

3.1.3. Protocol-accessible Nodes

The "ietf-tcp-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "tcp-client-grouping" populated with some data. This example shows a TCP-client configured to not connect via a proxy:

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-client">
  <remote-address>www.example.com</remote-address>
  <remote-port>443</remote-port>
  <local-address>0.0.0.0</local-address>
  <local-port>0</local-port>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-client>
```

This example shows a TCP-client configured to connect via a proxy:

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-client">
  <remote-address>www.example.com</remote-address>
  <remote-port>443</remote-port>
  <local-address>0.0.0.0</local-address>
  <local-port>0</local-port>
  <proxy-server>
    <socks5-parameters>
      <remote-address>proxy.my-domain.com</remote-address>
      <remote-port>1080</remote-port>
      <authentication-parameters>
        <username-password>
          <username>foobar</username>
          <cleartext-password>secret</cleartext-password>
        </username-password>
      </authentication-parameters>
    </socks5-parameters>
  </proxy-server>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-client>
```

3.3. YANG Module

The ietf-tcp-client YANG module references [RFC6991].

```
<CODE BEGINS> file "ietf-tcp-client@2022-03-07.yang"

module ietf-tcp-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-client";
  prefix tcpc;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

  import ietf-tcp-common {
    prefix tcpcmn;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group and the
     IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
               https://datatracker.ietf.org/wg/tcpm
     WG List:  NETCONF WG list <mailto:netconf@ietf.org>
               TCPM WG list <mailto:tcpm@ietf.org>
     Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
               Michael Scharf
               <mailto:michael.scharf@hs-esslingen.de>";

  description
    "This module defines reusable groupings for TCP clients that
     can be used as a basis for specific TCP client instances.

     Copyright (c) 2021 IETF Trust and the persons identified
     as authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Revised
```

```
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC DDDD
        (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

    revision 2022-03-07 {
      description
        "Initial version";
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    // Features

    feature local-binding-supported {
      description
        "Indicates that the server supports configuring local
         bindings (i.e., the local address and local port) for
         TCP clients.";
    }

    feature tcp-client-keepalives {
      description
        "Per socket TCP keepalive parameters are configurable for
         TCP clients on the server implementing this feature.";
    }

    feature proxy-connect {
      description
        "Proxy connection configuration is configurable for
         TCP clients on the server implementing this feature.";
    }

    feature socks5-gss-api {
      description
        "Indicates that the server supports authenticating
         using GSSAPI when initiating TCP connections via
         and SOCKS Version 5 proxy server.";
```

```
      reference
        "RFC 1928: SOCKS Protocol Version 5";
    }

    feature socks5-username-password {
      description
        "Indicates that the server supports authenticating using
         username/password when initiating TCP connections via
         and SOCKS Version 5 proxy server.";
      reference
        "RFC 1928: SOCKS Protocol Version 5";
    }

    // Groupings

    grouping tcp-client-grouping {
      description
        "A reusable grouping for configuring a TCP client.

        Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tcp-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      leaf remote-address {
        type inet:host;
        mandatory true;
        description
          "The IP address or hostname of the remote peer to
           establish a connection with.  If a domain name is
           configured, then the DNS resolution should happen on
           each connection attempt.  If the DNS resolution
           results in multiple IP addresses, the IP addresses
           are tried according to local preference order until
           a connection has been established or until all IP
           addresses have failed.";
      }
      leaf remote-port {
        type inet:port-number;
        default "0";
        description
          "The IP port number for the remote peer to establish a
           connection with.  An invalid default value (0) is used
           (instead of 'mandatory true') so that as application
```

```
            level data model may 'refine' it with an application
            specific default port number value.";
        }
        leaf local-address {
          if-feature "local-binding-supported";
          type inet:ip-address;
          description
            "The local IP address/interface (VRF?) to bind to for when
             connecting to the remote peer.  INADDR_ANY ('0.0.0.0') or
             INADDR6_ANY ('0:0:0:0:0:0:0:0' a.k.a. '::') MAY be used to
             explicitly indicate the implicit default, that the server
             can bind to any IPv4 or IPv6 addresses, respectively.";
        }
        leaf local-port {
          if-feature "local-binding-supported";
          type inet:port-number;
          default "0";
          description
            "The local IP port number to bind to for when connecting
             to the remote peer.  The port number '0', which is the
             default value, indicates that any available local port
             number may be used.";
        }
        container proxy-server {
          if-feature "proxy-connect";
          presence
            "Indicates that a proxy connection has been configured.
             Present so that the mandatory descendant nodes do not
             imply that this node must be configured.";
          choice proxy-type {
            mandatory true;
            description
              "Selects a proxy connection protocol.";
            case socks4 {
              container socks4-parameters {
                leaf remote-address {
                  type inet:ip-address;
                  mandatory true;
                  description
                    "The IP address of the proxy server.";
                }
                leaf remote-port {
                  type inet:port-number;
                  default "1080";
                  description
                    "The IP port number for the proxy server.";
                }
                description
```

```
            "Parameters for connecting to a TCP-based proxy
             server using the SOCKS4 protocol.";
          reference
            "SOCKS, Proceedings: 1992 Usenix Security Symposium.";
        }
      }
      case socks4a {
        container socks4a-parameters {
          leaf remote-address {
            type inet:host;
            mandatory true;
            description
              "The IP address or hostname of the proxy server.";
          }
          leaf remote-port {
            type inet:port-number;
            default "1080";
            description
              "The IP port number for the proxy server.";
          }
          description
            "Parameters for connecting to a TCP-based proxy
             server using the SOCKS4a protocol.";
          reference
            "SOCKS Proceedings:
               1992 Usenix Security Symposium.
             OpenSSH message:
               SOCKS 4A: A Simple Extension to SOCKS 4 Protocol
               https://www.openssh.com/txt/socks4a.protocol";
        }
      }
      case socks5 {
        container socks5-parameters {
          leaf remote-address {
            type inet:host;
            mandatory true;
            description
              "The IP address or hostname of the proxy server.";
          }
          leaf remote-port {
            type inet:port-number;
            default "1080";
            description
              "The IP port number for the proxy server.";
          }
          container authentication-parameters {
            presence
              "Indicates that an authentication mechanism
```

```
           has been configured.  Present so that the
           mandatory descendant nodes do not imply that
           this node must be configured.";
        description
          "A container for SOCKS Version 5 authentication
           mechanisms.

           A complete list of methods is defined at:
           https://www.iana.org/assignments/socks-methods
           /socks-methods.xhtml.";
        reference
          "RFC 1928: SOCKS Protocol Version 5";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst supported SOCKS Version 5
             authentication mechanisms.";
          case gss-api {
            if-feature "socks5-gss-api";
            container gss-api {
              description
                "Contains GSS-API configuration.  Defines
                 as an empty container to enable specific
                 GSS-API configuration to be augmented in
                 by future modules.";
              reference
                "RFC 1928: SOCKS Protocol Version 5
                 RFC 2743: Generic Security Service
                           Application Program Interface
                           Version 2, Update 1";
            }
          }
          case username-password {
            if-feature "socks5-username-password";
            container username-password {
              leaf username {
                type string;
                mandatory true;
                description
                  "The 'username' value to use for client
                   identification.";
              }
              uses ct:password-grouping {
                description
                  "The password to be used for client
                   authentication.";
              }
              description
```

```
                        "Contains Username/Password configuration.";
                      reference
                        "RFC 1929: Username/Password Authentication
                                   for SOCKS V5";
                    }
                  }
                }
              }
              description
                "Parameters for connecting to a TCP-based proxy server
                 using the SOCKS5 protocol.";
              reference
                "RFC 1928: SOCKS Protocol Version 5";
            }
          }
        }
        description
          "Proxy server settings.";
      }

      uses tcpcmn:tcp-common-grouping {
        augment "keepalives" {
          if-feature "tcp-client-keepalives";
          description
            "Add an if-feature statement so that implementations
             can choose to support TCP client keepalives.";
        }
      }
    }
  }
```

<CODE ENDS>

4. The "ietf-tcp-server" Module

This section defines a YANG 1.1 module called "ietf-tcp-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-tcp-server" module in terms of its features and groupings.

4.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tcp-server" module:

```
Features:
  +-- tcp-server-keepalives
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

4.1.2. Groupings

The "ietf-tcp-server" module defines the following "grouping" statement:

* tcp-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "tcp-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tcp-server-grouping" grouping:

```
  grouping tcp-server-grouping
    +-- local-address                 inet:ip-address
    +-- local-port?                   inet:port-number
    +---u tcpcmn:tcp-common-grouping
```

Comments:

* The "local-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* This grouping uses the "tcp-common-grouping" grouping discussed in Section 2.1.3.1.

4.1.3. Protocol-accessible Nodes

The "ietf-tcp-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

4.2. Example Usage

This section presents an example showing the "tcp-server-grouping" populated with some data.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-server">
  <local-address>10.20.30.40</local-address>
  <local-port>7777</local-port>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-server>
```

4.3. YANG Module

The ietf-tcp-server YANG module references [RFC6991].

```
<CODE BEGINS> file "ietf-tcp-server@2022-03-07.yang"

module ietf-tcp-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-server";
  prefix tcps;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-tcp-common {
    prefix tcpcmn;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group and the
     IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
               https://datatracker.ietf.org/wg/tcpm
```

```
        WG List:  NETCONF WG list <mailto:netconf@ietf.org>
                  TCPM WG list <mailto:tcpm@ietf.org>
        Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                  Michael Scharf
                  <mailto:michael.scharf@hs-esslingen.de>";

    description
      "This module defines reusable groupings for TCP servers that
       can be used as a basis for specific TCP server instances.

       Copyright (c) 2021 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

       Redistribution and use in source and binary forms, with
       or without modification, is permitted pursuant to, and
       subject to the license terms contained in, the Revised
       BSD License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC DDDD
       (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
       itself for full legal notices.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
       'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
       'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
       are to be interpreted as described in BCP 14 (RFC 2119)
       (RFC 8174) when, and only when, they appear in all
       capitals, as shown here.";

    revision 2022-03-07 {
      description
        "Initial version";
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    // Features

    feature tcp-server-keepalives {
      description
        "Per socket TCP keepalive parameters are configurable for
         TCP servers on the server implementing this feature.";
    }

    // Groupings
```

```
    grouping tcp-server-grouping {
      description
        "A reusable grouping for configuring a TCP server.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tcp-server-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";
      leaf local-address {
        type inet:ip-address;
        mandatory true;
        description
          "The local IP address to listen on for incoming
           TCP client connections.  INADDR_ANY (0.0.0.0) or
           INADDR6_ANY (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be
           used when the server is to listen on all IPv4 or
           IPv6 addresses, respectively.";
      }
      leaf local-port {
        type inet:port-number;
        default "0";
        description
          "The local port number to listen on for incoming TCP
           client connections.  An invalid default value (0)
           is used (instead of 'mandatory true') so that an
           application level data model may 'refine' it with
           an application specific default port number value.";
      }
      uses tcpcmn:tcp-common-grouping {
        augment "keepalives" {
          if-feature "tcp-server-keepalives";
          description
            "Add an if-feature statement so that implementations
             can choose to support TCP server keepalives.";
        }
      }
    }
  }

  <CODE ENDS>
```

5. Security Considerations

5.1. The "ietf-tcp-common" YANG Module

The "ietf-tcp-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. The "ietf-tcp-client" YANG Module

The "ietf-tcp-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "proxy-server/socks5-parameters/authentication-parameters/username-password/password" node:

  The cleartext "password" node defined in the "tcp-client-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Implementations are RECOMMENDED to implement the "local-binding-supported" feature for cryptographically-secure protocols, so as to enable more granular ingress/egress firewall rulebases. It is NOT RECOMMENDED to implement this feature for unsecure protocols, as per [RFC6056].

5.3. The "ietf-tcp-server" YANG Module

The "ietf-tcp-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-tcp-common
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tcp-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tcp-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

6.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-tcp-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-common
prefix:       tcpcmn
reference:    RFC DDDD

name:         ietf-tcp-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-client
prefix:       tcpc
reference:    RFC DDDD

name:         ietf-tcp-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-server
prefix:       tcps
reference:    RFC DDDD
```

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

7.2. Informative References

[I-D.ietf-netconf-crypto-types] Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[I-D.ietf-netconf-http-client-server] Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <https://www.rfc-editor.org/info/rfc6056>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Added 'local-binding-supported' feature to TCP-client model.

* Added 'keepalives-supported' feature to TCP-common model.

* Added 'external-endpoint-values' container and 'external-endpoints' feature to TCP-server model.

A.2. 01 to 02

* Removed the 'external-endpoint-values' container and 'external-endpoints' feature from the TCP-server model.

A.3. 02 to 03

* Moved the common model section to be before the client and server specific sections.

* Added sections "Model Scope" and "Usage Guidelines for Configuring TCP Keep-Alives" to the common model section.

A.4. 03 to 04

* Fixed a few typos.

A.5. 04 to 05

* Removed commented out "grouping tcp-system-grouping" statement kept for reviewers.

* Added a "Note to Reviewers" note to first page.

A.6. 05 to 06

* Added support for TCP proxies.

A.7. 06 to 07

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.8. 07 to 08

* Added missing IANA registration for "ietf-tcp-common"

* Added "mandatory true" for the "username" and "password" leafs

* Added an example of a TCP-client configured to connect via a proxy

* Fixed issues found by the SecDir review of the "keystore" draft.

* Updated the "ietf-tcp-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

A.9. 08 to 09

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.10. 09 to 10

* Updated Abstract and Intro to address comments by Tom Petch.

* Removed the "tcp-connection-grouping" grouping (now models use the "tcp-common-grouping" directly).

* Added XML-comment above examples explaining the reason for the unusual top-most element's presence.

* Added Securty Considerations section for the "local-binding-supported" feature.

* Replaced some hardcoded refs to <xref> elements.

* Fixed nits found by YANG Doctor reviews.

* Aligned modules with `pyang -f` formatting.

* Added an "Acknowledgements" secetion.

A.11. 10 to 11

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

A.12. 11 to 12

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)

* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Juergen Schoenwaelder, Ladislav Lhotka, Nick Hancock, and Tom Petch.

Authors' Addresses

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

Michael Scharf
Hochschule Esslingen - University of Applied Sciences
Email: michael.scharf@hs-esslingen.de

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-27

Abstract

This document defines three YANG 1.1 modules: the first defines features and groupings common to both TLS clients and TLS servers, the second defines a grouping for a generic TLS client, and the third defines a grouping for a generic TLS server.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* FFFF --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix B. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 4
  1.1. Relation to other RFCs 4
  1.2. Specification Language 6
  1.3. Adherence to the NMDA 6
  1.4. Conventions 6
2. The "ietf-tls-common" Module 7
  2.1. Data Model Overview 7
  2.2. Example Usage 9
  2.3. YANG Module 9
3. The "ietf-tls-client" Module 14
  3.1. Data Model Overview 14
  3.2. Example Usage 17
  3.3. YANG Module 21
4. The "ietf-tls-server" Module 33
  4.1. Data Model Overview 33

4.2. Example Usage 35
4.3. YANG Module 39
5. Security Considerations 51
5.1. The "iana-tls-cipher-suite-algs" Module 51
5.2. The "ietf-tls-common" YANG Module 51
5.3. The "ietf-tls-client" YANG Module 52
5.4. The "ietf-tls-server" YANG Module 53
6. IANA Considerations 53
6.1. The "IETF XML" Registry 53
6.2. The "YANG Module Names" Registry 54
6.3. The "iana-tls-cipher-suite-algs" Module 54
7. References 55
7.1. Normative References 55
7.2. Informative References 56
Appendix A. YANG Modules for IANA 59
A.1. Initial Module for the "TLS Cipher Suites" Registry . . . 59
A.1.1. Data Model Overview 59
A.1.2. Example Usage 60
A.1.3. YANG Module 61
Appendix B. Change Log 139
B.1. 00 to 01 139
B.2. 01 to 02 139
B.3. 02 to 03 139
B.4. 03 to 04 140
B.5. 04 to 05 140
B.6. 05 to 06 140
B.7. 06 to 07 140
B.8. 07 to 08 140
B.9. 08 to 09 141
B.10. 09 to 10 141
B.11. 10 to 11 141
B.12. 11 to 12 141
B.13. 12 to 13 142
B.14. 12 to 13 142
B.15. 13 to 14 142
B.16. 14 to 15 142
B.17. 15 to 16 142
B.18. 16 to 17 143
B.19. 17 to 18 143
B.20. 18 to 19 143
B.21. 19 to 20 144
B.22. 20 to 21 144
B.23. 21 to 22 144
B.24. 22 to 23 145
B.25. 23 to 24 145
B.26. 24 to 25 145
B.27. 25 to 26 145
B.28. 26 to 27 145

Acknowledgements 146
Contributors 146
Author's Address 146

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines features and groupings common to both TLS clients and TLS servers, the second defines a grouping for a generic TLS client, and the third defines a grouping for a generic TLS server.

Any version of TLS may be configured. TLS 1.0 [RFC2246] and TLS 1.1 [RFC4346] are historic and hence the YANG "feature" statements enabling them are marked "status obsolete". TLS 1.2 [RFC5246] is obsoleted by TLS 1.3 [RFC8446] but still in common use, and hence its "feature" statement is marked "status deprecated". All the feature statements for 1.0, 1.1, and 1.3 have "description" statements stating that it is NOT RECOMMENDED to enable obsolete protocol versions.

It is intended that the YANG groupings will be used by applications needing to configure TLS client and server protocol stacks. For instance, these groupings are used to help define the data model for HTTPS [RFC2818] and NETCONF over TLS [RFC7589] based clients and servers in [I-D.ietf-netconf-http-client-server] and [I-D.ietf-netconf-netconf-client-server] respectively.

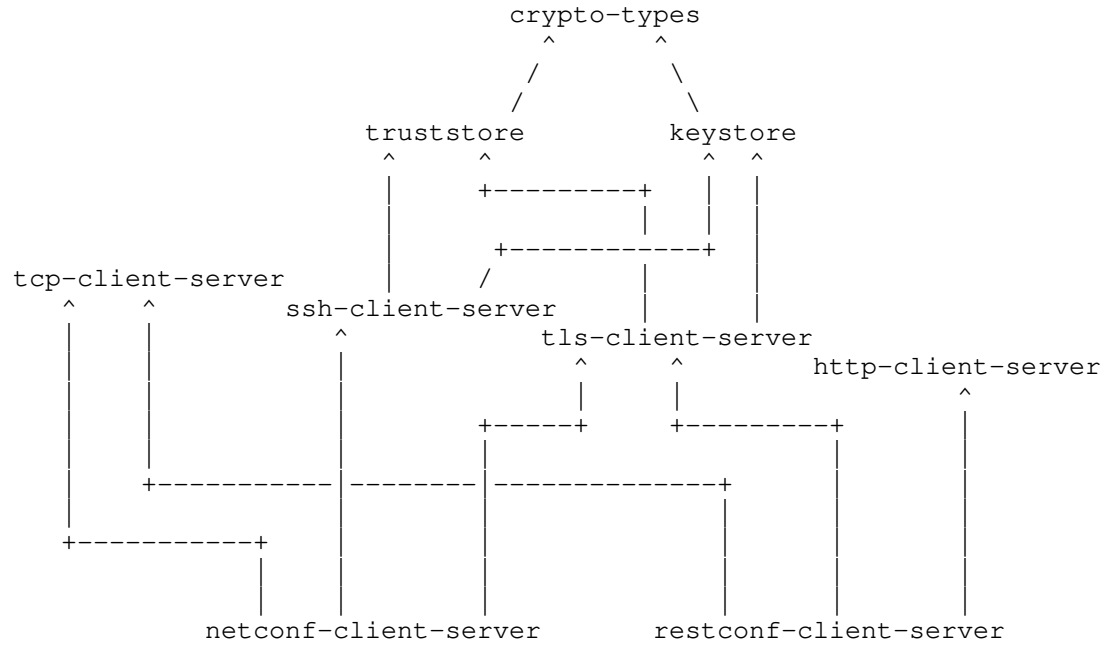
The client and server YANG modules in this document each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "tls-server-grouping" grouping for the TLS parts it provides, while adding data nodes for the TCP-level call-home configuration.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-tls-common" Module

The TLS common model presented in this section contains features and groupings common to both TLS clients and TLS servers. The "hello-params-grouping" grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for TLS clients and TLS servers that are capable of doing so and may serve to make TLS clients and TLS servers compliant with local security policies. This model supports both TLS1.2 [RFC5246] and TLS 1.3 [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the "hello-params-grouping" grouping should include three parameters for configuring its permitted TLS algorithms, which are: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Note that TLS1.2 only uses TLS Cipher Suites.

2.1. Data Model Overview

This section provides an overview of the "ietf-tls-common" module in terms of its features, identitiesm and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tls-common" module:

```
Features:
  +-- tls-1_0
  +-- tls-1_1
  +-- tls-1_2
  +-- tls-1_3
  +-- hello-params
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-tls-common" module:

```
Identities:
  +-- tls-version-base
     +-- tls-1.0
     +-- tls-1.1
     +-- tls-1.2
     +-- tls-1.3
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* The diagram shows that there are two base identities.
* One base identity is used to specific TLS versions, while the other is used to specify cipher-suites.
* These base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of things, rather than a specific thing.

2.1.3. Groupings

The "ietf-tls-common" module defines the following "grouping" statement:

* hello-params-grouping

This grouping is presented in the following subsection.

2.1.3.1. The "hello-params-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "hello-params-grouping" grouping:

```
  grouping hello-params-grouping
    +-- tls-versions
    |  +-- tls-version*   identityref
    +-- cipher-suites
       +-- cipher-suite*   identityref
```

Comments:

* This grouping is used by both the "tls-client-grouping" and the "tls-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.

* This grouping enables client and server configurations to specify the TLS versions and cipher suites that are to be used when establishing TLS sessions.

* The "cipher-suites" list is "ordered-by user".

2.1.4. Protocol-accessible Nodes

The "ietf-tls-common" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

2.2. Example Usage

This section shows how it would appear if the "hello-params-grouping" grouping were populated with some data.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<hello-params
   xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common"
   xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common"
   xmlns:tlscsa="urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-a\
lgs">
  <tls-versions>
    <tls-version>tlscmn:tls-1.1</tls-version>
    <tls-version>tlscmn:tls-1.2</tls-version>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>tlscsa:tls-ecdhe-ecdsa-with-aes-256-cbc-sha</ciphe\
r-suite>
    <cipher-suite>tlscsa:tls-dhe-rsa-with-aes-128-cbc-sha256</cipher\
-suite>
    <cipher-suite>tlscsa:tls-rsa-with-3des-ede-cbc-sha</cipher-suite>
  </cipher-suites>
</hello-params>
```

2.3. YANG Module

This YANG module has a normative references to [RFC4346], [RFC5288], [RFC5289], [RFC8422], and FIPS PUB 180-4.

This YANG module has a informative references to [RFC2246], [RFC4346], [RFC5246], and [RFC8446].

<CODE BEGINS> file "ietf-tls-common@2022-03-07.yang"

```
   module ietf-tls-common {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
     prefix tlscmn;

     import iana-tls-cipher-suite-algs {
       prefix tlscsa;
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and SSH Servers";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";

     contact
       "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        WG Web:   https://datatracker.ietf.org/wg/netconf
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>
        Author:   Jeff Hartley <mailto:jeff.hartley@commscope.com>";

      description
       "This module defines a common features and groupings for
        Transport Layer Security (TLS).

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

     revision 2022-03-07 {
       description
```

```
        "Initial version";
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    // Features

    feature tls10 {
      status "obsolete";
      description
        "TLS Protocol Version 1.0 is supported.  TLS 1.0 is obsolete
         and thus it is NOT RECOMMENDED to enable this feature.";
      reference
        "RFC 2246: The TLS Protocol Version 1.0";
    }

    feature tls11 {
      status "obsolete";
      description
        "TLS Protocol Version 1.1 is supported.  TLS 1.1 is obsolete
         and thus it is NOT RECOMMENDED to enable this feature.";
      reference
        "RFC 4346: The Transport Layer Security (TLS) Protocol
                   Version 1.1";
    }

    feature tls12 {
      status "deprecated";
      description
        "TLS Protocol Version 1.2 is supported  TLS 1.2 is obsolete
         and thus it is NOT RECOMMENDED to enable this feature.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    feature tls13 {
      description
        "TLS Protocol Version 1.3 is supported.";
      reference
        "RFC 8446: The Transport Layer Security (TLS)
                   Protocol Version 1.3";
    }

    feature hello-params {
      description
        "TLS hello message parameters are configurable.";
    }
```

```
  // Identities

  identity tls-version-base {
    description
      "Base identity used to identify TLS protocol versions.";
  }

  identity tls10 {
    if-feature "tls10";
    base tls-version-base;
    status "obsolete";
    description
      "TLS Protocol Version 1.0.";
    reference
      "RFC 2246: The TLS Protocol Version 1.0";
  }

  identity tls11 {
    if-feature "tls11";
    base tls-version-base;
    status "obsolete";
    description
      "TLS Protocol Version 1.1.";
    reference
      "RFC 4346: The Transport Layer Security (TLS) Protocol
                 Version 1.1";
  }

  identity tls12 {
    if-feature "tls12";
    base tls-version-base;
    status "deprecated";
    description
      "TLS Protocol Version 1.2.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
                 Version 1.2";
  }

  identity tls13 {
    if-feature "tls13";
    base tls-version-base;
    description
      "TLS Protocol Version 1.3.";
    reference
      "RFC 8446: The Transport Layer Security (TLS)
                 Protocol Version 1.3";
  }
```

```
  typedef epsk-supported-hash {
    type enumeration {
      enum sha-256 {
        description
          "The SHA-256 Hash.";
      }
      enum sha-384 {
        description
          "The SHA-384 Hash.";
      }
    }
    description
      "As per Section 4.2.11 of RFC 8446, the hash algorithm
       supported by an instance of an External Pre-Shared
       Key (EPSK).";
    reference
      "RFC 8446: The Transport Layer Security (TLS)
                 Protocol Version 1.3
       I-D.ietf-tls-external-psk-importer: Importing
                 External PSKs for TLS
       I-D.ietf-tls-external-psk-guidance: Guidance
                 for External PSK Usage in TLS";
  }


  // Groupings

  grouping hello-params-grouping {
    description
      "A reusable grouping for TLS hello message parameters.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
                 Version 1.2
       RFC 8446: The Transport Layer Security (TLS) Protocol
                 Version 1.3";
    container tls-versions {
      description
        "Parameters regarding TLS versions.";
      leaf-list tls-version {
        type identityref {
          base tls-version-base;
        }
        description
          "Acceptable TLS protocol versions.

           If this leaf-list is not configured (has zero elements)
           the acceptable TLS protocol versions are implementation-
           defined.";
```

```
          }
        }
        container cipher-suites {
          description
            "Parameters regarding cipher suites.";
          leaf-list cipher-suite {
            type identityref {
              base tlscsa:cipher-suite-alg-base;
            }
            ordered-by user;
            description
              "Acceptable cipher suites in order of descending
               preference.  The configured host key algorithms should
               be compatible with the algorithm used by the configured
               private key.  Please see Section 5 of RFC FFFF for
               valid combinations.

               If this leaf-list is not configured (has zero elements)
               the acceptable cipher suites are implementation-
               defined.";
            reference
              "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
          }
        }
      } // hello-params-grouping

   }

   <CODE ENDS>
```

3. The "ietf-tls-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-tls-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-tls-client" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tls-client" module:

```
   Features:
     +-- tls-client-keepalives
     +-- client-ident-x509-cert
     +-- client-ident-raw-public-key
     +-- client-ident-psk
     +-- server-auth-x509-cert
     +-- server-auth-raw-public-key
     +-- server-auth-psk
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

3.1.2. Groupings

The "ietf-tls-client" module defines the following "grouping" statement:

* tls-client-grouping

This grouping is presented in the following subsection.

3.1.2.1. The "tls-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tls-client-grouping" grouping:

```
   ============== NOTE: '\' line wrapping per RFC 8792 ================

     grouping tls-client-grouping
       +-- client-identity!
       |  +-- (auth-type)
       |     +--:(certificate) {client-ident-x509-cert}?
       |     |  +-- certificate
       |     |     +---u ks:local-or-keystore-end-entity-cert-with-key-\
   grouping
       |     +--:(raw-public-key) {client-ident-raw-public-key}?
       |     |  +-- raw-private-key
       |     |     +---u ks:local-or-keystore-asymmetric-key-grouping
       |     +--:(tls12-psk) {client-ident-tls12-psk}?
       |     |  +-- tls12-psk
       |     |     +---u ks:local-or-keystore-symmetric-key-grouping
       |     |     +-- id?
       |     |              string
       |     +--:(tls13-epsk) {client-ident-tls13-epsk}?
       |        +-- tls13-epsk
       |           +---u ks:local-or-keystore-symmetric-key-grouping
       |           +-- external-identity
       |           |       string
       |           +-- hash
       |           |       tlscmn:epsk-supported-hash
       |           +-- context?
       |           |       string
       |           +-- target-protocol?
       |           |       uint16
       |           +-- target-kdf?
       |                   uint16
       +-- server-authentication
       |  +-- ca-certs! {server-auth-x509-cert}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- ee-certs! {server-auth-x509-cert}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- raw-public-keys! {server-auth-raw-public-key}?
       |  |  +---u ts:local-or-truststore-public-keys-grouping
       |  +-- tls12-psks?        empty {server-auth-tls12-psk}?
       |  +-- tls13-epsks?       empty {server-auth-tls13-epsk}?
       +-- hello-params {tlscmn:hello-params}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives {tls-client-keepalives}?
          +-- peer-allowed-to-send?   empty
          +-- test-peer-aliveness!
             +-- max-wait?       uint16
             +-- max-attempts?   uint8

   Comments:
```

* The "client-identity" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures identity credentials, each enabled by a "feature" statement defined in Section 3.1.1.

* The "server-authentication" node configures trust anchors for authenticating the TLS server, with each option enabled by a "feature" statement.

* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the TLS server. The aliveness-test occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-tls-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "tls-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the client identity and server authentication:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tls-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <!-- how this client will authenticate itself to the server -->
    <client-identity>
        <certificate>
            <local-definition>
                <public-key-format>ct:subject-public-key-info-format\
</public-key-format>
                <public-key>BASE64VALUE=</public-key>
                <private-key-format>ct:rsa-private-key-format</priva\
te-key-format>
                <cleartext-private-key>BASE64VALUE=</cleartext-priva\
te-key>
                <cert-data>BASE64VALUE=</cert-data>
            </local-definition>
        </certificate>
        <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
TIME
        <raw-private-key>
         <local-definition>
            <public-key-format>ct:subject-public-key-info-format</pu\
blic-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>ct:rsa-private-key-format</private-k\
ey-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-k\
ey>
          </local-definition>
        </raw-private-key>
        -->
        <!-- USE ONLY ONE AT A TIME
        <tls12-psk>
            <local-definition>
                <key-format>ct:octet-string-key-format</key-format>
                <cleartext-key>BASE64VALUE=</cleartext-key>
            </local-definition>
            <id>example_id_string</id>
        </tls12-psk>
        -->
```

```
            <tls13-epsk>
                <local-definition>
                    <key-format>ct:octet-string-key-format</key-format>
                    <cleartext-key>BASE64VALUE=</cleartext-key>
                </local-definition>
                <external-identity>example_external_id</external-identit\
   y>
                <hash>sha-256</hash>
                <context>example_context_string</context>
                <target-protocol>8443</target-protocol>
                <target-kdf>12345</target-kdf>
            </tls13-epsk>
        </client-identity>
        <!-- which certificates will this client trust -->
        <server-authentication>
            <ca-certs>
                <local-definition>
                    <certificate>
                        <name>Server Cert Issuer #1</name>
                        <cert-data>BASE64VALUE=</cert-data>
                    </certificate>
                    <certificate>
                        <name>Server Cert Issuer #2</name>
                        <cert-data>BASE64VALUE=</cert-data>
                    </certificate>
                </local-definition>
            </ca-certs>
            <ee-certs>
                <local-definition>
                    <certificate>
                        <name>My Application #1</name>
                        <cert-data>BASE64VALUE=</cert-data>
                    </certificate>
                    <certificate>
                        <name>My Application #2</name>
                        <cert-data>BASE64VALUE=</cert-data>
                    </certificate>
                </local-definition>
            </ee-certs>
            <raw-public-keys>
                <local-definition>
                    <public-key>
                        <name>corp-fw1</name>
                        <public-key-format>ct:subject-public-key-info-fo\
   rmat</public-key-format>
                        <public-key>BASE64VALUE=</public-key>
                    </public-key>
                    <public-key>
```

```
                    <name>corp-fw1</name>
                    <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                    <public-key>BASE64VALUE=</public-key>
                </public-key>
            </local-definition>
        </raw-public-keys>
        <tls12-psks/>
        <tls13-epsks/>
    </server-authentication>
    <keepalives>
        <test-peer-aliveness>
            <max-wait>30</max-wait>
            <max-attempts>3</max-attempts>
        </test-peer-aliveness>
    </keepalives>
</tls-client>
```

The following configuration example uses keystore-references for the client identity and truststore-references for server authentication: from the keystore:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tls-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client">
    <!-- how this client will authenticate itself to the server -->
    <client-identity>
        <certificate>
            <keystore-reference>
                <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
        </certificate>
        <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
TIME
        <raw-private-key>
          <keystore-reference>raw-private-key</keystore-reference>
        </raw-private-key>
        -->
        <!-- USE ONLY ONE AT A TIME
        <tls12-psk>
            <keystore-reference>encrypted-symmetric-key</keystore-re\
ference>
            <id>example_id_string</id>
        </tls12-psk>
```

```
            -->
            <tls13-epsk>
                <keystore-reference>encrypted-symmetric-key</keystore-re\
   ference>
                <external-identity>example_external_id</external-identit\
   y>
                <hash>sha-256</hash>
                <context>example_context_string</context>
                <target-protocol>8443</target-protocol>
                <target-kdf>12345</target-kdf>
            </tls13-epsk>
        </client-identity>
        <!-- which certificates will this client trust -->
        <server-authentication>
            <ca-certs>
                <truststore-reference>trusted-server-ca-certs</truststor\
   e-reference>
            </ca-certs>
            <ee-certs>
                <truststore-reference>trusted-server-ee-certs</truststor\
   e-reference>
            </ee-certs>
            <raw-public-keys>
                <truststore-reference>Raw Public Keys for TLS Servers</t\
   ruststore-reference>
            </raw-public-keys>
            <tls12-psks/>
            <tls13-epsks/>
        </server-authentication>
        <keepalives>
            <test-peer-aliveness>
                <max-wait>30</max-wait>
                <max-attempts>3</max-attempts>
            </test-peer-aliveness>
        </keepalives>
   </tls-client>
```

3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], and Informative references to [RFC5246], [RFC8446], [I-D.ietf-tls-external-psk-importer] and [I-D.ietf-tls-external-psk-guidance].

<CODE BEGINS> file "ietf-tls-client@2022-03-07.yang"

```
   module ietf-tls-client {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
     prefix tlsc;

     import ietf-netconf-acm {
       prefix nacm;
       reference
         "RFC 8341: Network Configuration Access Control Model";
     }

     import ietf-crypto-types {
       prefix ct;
       reference
         "RFC AAAA: YANG Data Types and Groupings for Cryptography";
     }

     import ietf-truststore {
       prefix ts;
       reference
         "RFC BBBB: A YANG Data Model for a Truststore";
     }

     import ietf-keystore {
       prefix ks;
       reference
         "RFC CCCC: A YANG Data Model for a Keystore";
     }

     import ietf-tls-common {
       prefix tlscmn;
       revision-date 2022-03-07; // stable grouping definitions
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";

     contact
       "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        WG Web:   https://datatracker.ietf.org/wg/netconf
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>
        Author:   Jeff Hartley <mailto:jeff.hartley@commscope.com>";

     description
       "This module defines reusable groupings for TLS clients that
```

```
        can be used as a basis for specific TLS client instances.

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Features

      feature tls-client-keepalives {
        description
          "Per socket TLS keepalive parameters are configurable for
           TLS clients on the server implementing this feature.";
      }

      feature client-ident-x509-cert {
        description
          "Indicates that the client supports identifying itself
           using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }
```

```
    feature client-ident-raw-public-key {
      description
        "Indicates that the client supports identifying itself
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature client-ident-tls12-psk {
      description
        "Indicates that the client supports identifying itself
         using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature client-ident-tls13-epsk {
      description
        "Indicates that the client supports identifying itself
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    feature server-auth-x509-cert {
      description
        "Indicates that the client supports authenticating servers
         using X.509 certificates.";
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile";
    }

    feature server-auth-raw-public-key {
      description
        "Indicates that the client supports authenticating servers
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }
```

```
    feature server-auth-tls12-psk {
      description
        "Indicates that the client supports authenticating servers
         using PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature server-auth-tls13-epsk {
      description
        "Indicates that the client supports authenticating servers
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    // Groupings

    grouping tls-client-grouping {
      description
        "A reusable grouping for configuring a TLS client without
         any consideration for how an underlying TCP session is
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tls-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container client-identity {
        nacm:default-deny-write;
        presence
          "Indicates that a TLS-level client identity has been
           configured.  This statement is present so the mandatory
           descendant do not imply that this node must be configured.";
        description
          "Identity credentials the TLS client MAY present when
           establishing a connection to a TLS server.  If not
           configured, then client authentication is presumed to
           occur a protocol layer above TLS.  When configured,
           and requested by the TLS server when establishing a
```

```
           TLS session, these credentials are passed in the
           Certificate message defined in Section 7.4.2 of
           RFC 5246 and Section 4.4.2 in RFC 8446.";
        reference
          "RFC 5246: The Transport Layer Security (TLS)
                     Protocol Version 1.2
           RFC 8446: The Transport Layer Security (TLS)
                     Protocol Version 1.3
           RFC CCCC: A YANG Data Model for a Keystore";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst authentication types, of which one must
             be enabled (via its associated 'feature') and selected.";
          case certificate {
            if-feature "client-ident-x509-cert";
            container certificate {
              description
                "Specifies the client identity using a certificate.";
              uses
                ks:local-or-keystore-end-entity-cert-with-key-grouping{
                refine "local-or-keystore/local/local-definition" {
                  must 'public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
                }
                refine "local-or-keystore/keystore/keystore-reference"
                     + "/asymmetric-key" {
                  must 'deref(.)/../ks:public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
                }
              }
            }
          }
          case raw-public-key {
            if-feature "client-ident-raw-public-key";
            container raw-private-key {
              description
                "Specifies the client identity using a raw
                 private key.";
              uses ks:local-or-keystore-asymmetric-key-grouping {
                refine "local-or-keystore/local/local-definition" {
                  must 'public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
                }
                refine "local-or-keystore/keystore"
                     + "/keystore-reference" {
                  must 'deref(.)/../ks:public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
```

```
            }
          }
        }
      }
      case tls12-psk {
        if-feature "client-ident-tls12-psk";
        container tls12-psk {
          description
            "Specifies the client identity using a PSK (pre-shared
             or pairwise-symmetric key).";
          uses ks:local-or-keystore-symmetric-key-grouping;
          leaf id {
            type string;
            description
              "The key 'psk_identity' value used in the TLS
               'ClientKeyExchange' message.";
            reference
              "RFC 4279: Pre-Shared Key Ciphersuites for
                         Transport Layer Security (TLS)";
          }
        }
      }
      case tls13-epsk {
        if-feature "client-ident-tls13-epsk";
        container tls13-epsk {
          description
            "An External Pre-Shared Key (EPSK) is established
            or provisioned out-of-band, i.e., not from a TLS
            connection.  An EPSK is a tuple of (Base Key,
            External Identity, Hash).  External PSKs MUST NOT
            be imported for (D)TLS 1.2 or prior versions.  When
            PSKs are provisioned out of band, the PSK identity
            and the KDF hash algorithm to be used with the PSK
            MUST also be provisioned.

            The structure of this container is designed
            to satisfy the requirements of RFC 8446
            Section 4.2.11, the recommendations from I-D
            ietf-tls-external-psk-guidance Section 6,
            and the EPSK input fields detailed in I-D
            draft-ietf-tls-external-psk-importer
            Section 3.1.  The base-key is based upon
            ks:local-or-keystore-symmetric-key-grouping
            in order to provide users with flexible and
            secure storage options.";
          reference
            "RFC 8446: The Transport Layer Security (TLS)
                       Protocol Version 1.3
```

```
                I-D.ietf-tls-external-psk-importer:
                          Importing External PSKs for TLS
                I-D.ietf-tls-external-psk-guidance:
                          Guidance for External PSK Usage in TLS";
              uses ks:local-or-keystore-symmetric-key-grouping;
              leaf external-identity {
                type string;
                mandatory true;
                description
                  "As per Section 4.2.11 of RFC 8446, and Section 4.1
                   of I-D. ietf-tls-external-psk-guidance:
                   A sequence of bytes used to identify an EPSK. A
                   label for a pre-shared key established externally.";
                reference
                  "RFC 8446: The Transport Layer Security (TLS)
                             Protocol Version 1.3
                   I-D.ietf-tls-external-psk-guidance:
                             Guidance for External PSK Usage in TLS";
              }
              leaf hash {
                type tlscmn:epsk-supported-hash;
                mandatory true;
                description
                  "As per Section 4.2.11 of RFC 8446, for externally
                   established PSKs, the Hash algorithm MUST be set
                   when the PSK is established or default to SHA-256
                   if no such algorithm is defined.  The server MUST
                   ensure that it selects a compatible PSK (if any)
                   and cipher suite.  Each PSK MUST only be used with
                   a single hash function.";
                reference
                  "RFC 8446: The Transport Layer Security (TLS)
                             Protocol Version 1.3";
              }
              leaf context {
                type string;
                description
                  "As per Section 4.1 of I-D.
                   ietf-tls-external-psk-guidance: Context may include
                   information about peer roles or identities to
                   mitigate Selfie-style reflection attacks [Selfie].
                   If the EPSK is a key derived from some other
                   protocol or sequence of protocols, context
                   MUST include a channel binding for the deriving
                   protocols [RFC5056].  The details of this binding
                   are protocol specific.";
                reference
                  "I-D.ietf-tls-external-psk-importer:
```

```
                          Importing External PSKs for TLS
                    I-D.ietf-tls-external-psk-guidance:
                          Guidance for External PSK Usage in TLS";
              }
              leaf target-protocol {
                type uint16;
                description
                  "As per Section 3.1 of I-D.
                   ietf-tls-external-psk-guidance:
                   The protocol for which a PSK is imported for use.";
                reference
                  "I-D.ietf-tls-external-psk-importer:
                          Importing External PSKs for TLS";
              }
              leaf target-kdf {
                type uint16;
                description
                  "As per Section 3.1 of I-D.
                   ietf-tls-external-psk-guidance:
                   The specific Key Derivation Function (KDF) for which
                   a PSK is imported for use.";
                reference
                  "I-D.ietf-tls-external-psk-importer:
                          Importing External PSKs for TLS";
              }
            }
          }
        }
      } // container client-identity

      container server-authentication {
        nacm:default-deny-write;
        must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
          or tls13-epsks';
        description
          "Specifies how the TLS client can authenticate TLS servers.
           Any combination of credentials is additive and unordered.

           Note that no configuration is required for PSK (pre-shared
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../client-
           identity' node.";
        container ca-certs {
          if-feature "server-auth-x509-cert";
          presence
            "Indicates that CA certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
```

```
          description
            "A set of certificate authority (CA) certificates used by
             the TLS client to authenticate TLS server certificates.
             A server certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "server-auth-x509-cert";
          presence
            "Indicates that EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of server certificates (i.e., end entity
             certificates) used by the TLS client to authenticate
             certificates presented by TLS servers.  A server
             certificate is authenticated if it is an exact
             match to a configured server certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container raw-public-keys {
          if-feature "server-auth-raw-public-key";
          presence
            "Indicates that raw public keys have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of raw public keys used by the TLS client to
             authenticate raw public keys presented by the TLS
             server.  A raw public key is authenticated if it
             is an exact match to a configured raw public key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine "local-or-truststore/local/local-definition"
                 + "/public-key" {
              must 'public-key-format'
                 + ' = "ct:subject-public-key-info-format"';
            }
            refine "local-or-truststore/truststore"
                 + "/truststore-reference" {
              must 'deref(.)/../*/ts:public-key-format'
                 + ' = "ct:subject-public-key-info-format"';
```

```
            }
          }
        }
        leaf tls12-psks {
          if-feature "server-auth-tls12-psk";
          type empty;
          description
            "Indicates that the TLS client can authenticate TLS servers
             using configure PSKs (pre-shared or pairwise-symmetric
             keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'client-identity'
             node.";
        }
        leaf tls13-epsks {
          if-feature "server-auth-tls13-epsk";
          type empty;
          description
            "Indicates that the TLS client can authenticate TLS servers
             using configured external PSKs (pre-shared keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'client-identity'
             node.";
        }
      } // container server-authentication

      container hello-params {
        nacm:default-deny-write;
        if-feature "tlscmn:hello-params";
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      } // container hello-params

      container keepalives {
        nacm:default-deny-write;
        if-feature "tls-client-keepalives";
        description
          "Configures the keepalive policy for the TLS client.";
        leaf peer-allowed-to-send {
          type empty;
          description
            "Indicates that the remote TLS server is allowed to send
             HeartbeatRequest messages, as defined by RFC 6520
             to this TLS client.";
          reference
```

```
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
        }
        container test-peer-aliveness {
          presence
            "Indicates that the TLS client proactively tests the
             aliveness of the remote TLS server.";
          description
            "Configures the keep-alive policy to proactively test
             the aliveness of the TLS server.  An unresponsive
             TLS server is dropped after approximately max-wait
             * max-attempts seconds.  The TLS client MUST send
             HeartbeatRequest messages, as defined by RFC 6520.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "30";
            description
              "Sets the amount of time in seconds after which if
               no data has been received from the TLS server, a
               TLS-level message will be sent to test the
               aliveness of the TLS server.";
          }
          leaf max-attempts {
            type uint8;
            default "3";
            description
              "Sets the maximum number of sequential keep-alive
               messages that can fail to obtain a response from
               the TLS server before assuming the TLS server is
               no longer alive.";
          }
        }
      }
    } // grouping tls-client-grouping

  }

  <CODE ENDS>
```

4. The "ietf-tls-server" Module

This section defines a YANG 1.1 module called "ietf-tls-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-tls-server" module in terms of its features and groupings.

4.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tls-server" module:

```
Features:
  +-- tls-server-keepalives
  +-- server-ident-x509-cert
  +-- server-ident-raw-public-key
  +-- server-ident-psk
  +-- client-auth-supported
  +-- client-auth-x509-cert
  +-- client-auth-raw-public-key
  +-- client-auth-psk
```

> The diagram above uses syntax that is similar to but not defined in [RFC8340].

4.1.2. Groupings

The "ietf-tls-server" module defines the following "grouping" statement:

* tls-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "tls-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tls-server-grouping" grouping:

```
   ============== NOTE: '\' line wrapping per RFC 8792 ================

     grouping tls-server-grouping
       +-- server-identity
       |  +-- (auth-type)
       |     +--:(certificate) {server-ident-x509-cert}?
       |     |  +-- certificate
       |     |     +---u ks:local-or-keystore-end-entity-cert-with-key-\
   grouping
       |     +--:(raw-private-key) {server-ident-raw-public-key}?
       |     |  +-- raw-private-key
       |     |     +---u ks:local-or-keystore-asymmetric-key-grouping
       |     +--:(tls12-psk) {server-ident-tls12-psk}?
       |     |  +-- tls12-psk
       |     |     +---u ks:local-or-keystore-symmetric-key-grouping
       |     |     +-- id_hint?
       |     |             string
       |     +--:(tls13-epsk) {server-ident-tls13-epsk}?
       |        +-- tls13-epsk
       |           +---u ks:local-or-keystore-symmetric-key-grouping
       |           +-- external-identity
       |           |       string
       |           +-- hash
       |           |       tlscmn:epsk-supported-hash
       |           +-- context?
       |           |       string
       |           +-- target-protocol?
       |           |       uint16
       |           +-- target-kdf?
       |                   uint16
       +-- client-authentication! {client-auth-supported}?
       |  +-- ca-certs! {client-auth-x509-cert}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- ee-certs! {client-auth-x509-cert}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- raw-public-keys! {client-auth-raw-public-key}?
       |  |  +---u ts:local-or-truststore-public-keys-grouping
       |  +-- tls12-psks?        empty {client-auth-tls12-psk}?
       |  +-- tls13-epsks?       empty {client-auth-tls13-epsk}?
       +-- hello-params {tlscmn:hello-params}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives {tls-server-keepalives}?
          +-- peer-allowed-to-send?   empty
          +-- test-peer-aliveness!
             +-- max-wait?       uint16
             +-- max-attempts?   uint8

   Comments:
```

* The "server-identity" node configures identity credentials, each of which is enabled by a "feature".

* The "client-authentication" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures trust anchors for authenticating the TLS client, with each option enabled by a "feature" statement.

* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a flag enabling the TLS client to test the aliveness of the TLS server, as well as a "presence" container for testing the aliveness of the TLSi client. The aliveness-tests occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

4.1.3. Protocol-accessible Nodes

The "ietf-tls-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

4.2. Example Usage

This section presents two examples showing the "tls-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of

[I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the server identity and client authentication:

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tls-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <!-- how this server will authenticate itself to the client -->
    <server-identity>
        <certificate>
            <local-definition>
                <public-key-format>ct:subject-public-key-info-format\
</public-key-format>
                <public-key>BASE64VALUE=</public-key>
                <private-key-format>ct:rsa-private-key-format</priva\
te-key-format>
                <cleartext-private-key>BASE64VALUE=</cleartext-priva\
te-key>
                <cert-data>BASE64VALUE=</cert-data>
            </local-definition>
        </certificate>
        <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
TIME
        <raw-private-key>
          <local-definition>
            <public-key-format>ct:subject-public-key-info-format</pu\
blic-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>ct:rsa-private-key-format</private-k\
ey-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-k\
ey>
          </local-definition>
        </raw-private-key>
        -->
        <!-- USE ONLY ONE AT A TIME
        <tls12-psk>
            <local-definition>
                <key-format>ct:octet-string-key-format</key-format>
                <cleartext-key>BASE64VALUE=</cleartext-key>
            </local-definition>
            <id_hint>example_id_hint</id_hint>
```

```
          </tls12-psk>
          -->
          <tls13-epsk>
              <local-definition>
                  <key-format>ct:octet-string-key-format</key-format>
                  <cleartext-key>BASE64VALUE=</cleartext-key>
              </local-definition>
              <external-identity>example_external_id</external-identit\
  y>
              <hash>sha-256</hash>
              <context>example_context_string</context>
              <target-protocol>8443</target-protocol>
              <target-kdf>12345</target-kdf>
          </tls13-epsk>
      </server-identity>
      <!-- which certificates will this server trust -->
      <client-authentication>
          <ca-certs>
              <local-definition>
                  <certificate>
                      <name>Identity Cert Issuer #1</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
                  <certificate>
                      <name>Identity Cert Issuer #2</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
              </local-definition>
          </ca-certs>
          <ee-certs>
              <local-definition>
                  <certificate>
                      <name>Application #1</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
                  <certificate>
                      <name>Application #2</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
              </local-definition>
          </ee-certs>
          <raw-public-keys>
              <local-definition>
                  <public-key>
                      <name>User A</name>
                      <public-key-format>ct:subject-public-key-info-fo\
  rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
```

```
                  </public-key>
                  <public-key>
                      <name>User B</name>
                      <public-key-format>ct:subject-public-key-info-fo\
   rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
              </local-definition>
          </raw-public-keys>
          <tls12-psks/>
          <tls13-epsks/>
      </client-authentication>
      <keepalives>
          <peer-allowed-to-send/>
      </keepalives>
   </tls-server>
```

The following configuration example uses keystore-references for the server identity and truststore-references for client authentication: from the keystore:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">
       <!-- how this server will authenticate itself to the client -->
       <server-identity>
           <certificate>
               <keystore-reference>
                   <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                   <certificate>ex-rsa-cert</certificate>
               </keystore-reference>
           </certificate>
           <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
   TIME
           -->
           <!-- USE ONLY ONE AT A TIME
           <raw-private-key>
             <keystore-reference>raw-private-key</keystore-reference>
           </raw-private-key>
           -->
           <!-- USE ONLY ONE AT A TIME
           <tls12-psk>
               <keystore-reference>encrypted-symmetric-key</keystore-re\
   ference>
               <id_hint>example_id_hint</id_hint>
```

```
            </tls12-psk>
            -->
            <tls13-epsk>
                <keystore-reference>encrypted-symmetric-key</keystore-re\
   ference>
                <external-identity>example_external_id</external-identit\
   y>
                <hash>sha-256</hash>
                <context>example_context_string</context>
                <target-protocol>8443</target-protocol>
                <target-kdf>12345</target-kdf>
            </tls13-epsk>
        </server-identity>
        <!-- which certificates will this server trust -->
        <client-authentication>
            <ca-certs>
                <truststore-reference>trusted-client-ca-certs</truststor\
   e-reference>
            </ca-certs>
            <ee-certs>
                <truststore-reference>trusted-client-ee-certs</truststor\
   e-reference>
            </ee-certs>
            <raw-public-keys>
                <truststore-reference>Raw Public Keys for TLS Clients</t\
   ruststore-reference>
            </raw-public-keys>
            <tls12-psks/>
            <tls13-epsks/>
        </client-authentication>
        <keepalives>
            <peer-allowed-to-send/>
        </keepalives>
   </tls-server>
```

4.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], and Informative references to [RFC5246], [RFC8446], [I-D.ietf-tls-external-psk-importer] and [I-D.ietf-tls-external-psk-guidance].

<CODE BEGINS> file "ietf-tls-server@2022-03-07.yang"

```
   module ietf-tls-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
     prefix tlss;

     import ietf-netconf-acm {
       prefix nacm;
       reference
         "RFC 8341: Network Configuration Access Control Model";
     }

     import ietf-crypto-types {
       prefix ct;
       reference
         "RFC AAAA: YANG Data Types and Groupings for Cryptography";
     }

     import ietf-truststore {
       prefix ts;
       reference
         "RFC BBBB: A YANG Data Model for a Truststore";
     }

     import ietf-keystore {
       prefix ks;
       reference
         "RFC CCCC: A YANG Data Model for a Keystore";
     }

     import ietf-tls-common {
       prefix tlscmn;
       revision-date 2022-03-07; // stable grouping definitions
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";

     contact
       "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        WG Web:   https://datatracker.ietf.org/wg/netconf
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>
        Author:   Jeff Hartley <mailto:jeff.hartley@commscope.com>";

     description
       "This module defines reusable groupings for TLS servers that
```

```
        can be used as a basis for specific TLS server instances.

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

      revision 2022-03-07 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Features

      feature tls-server-keepalives {
        description
          "Per socket TLS keepalive parameters are configurable for
           TLS servers on the server implementing this feature.";
      }

      feature server-ident-x509-cert {
        description
          "Indicates that the server supports identifying itself
           using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }
```

```
    feature server-ident-raw-public-key {
      description
        "Indicates that the server supports identifying itself
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature server-ident-tls12-psk {
      description
        "Indicates that the server supports identifying itself
         using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature server-ident-tls13-epsk {
      description
        "Indicates that the server supports identifying itself
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    feature client-auth-supported {
      description
        "Indicates that the configuration for how to authenticate
         clients can be configured herein.  TLS-level client
         authentication may not be needed when client authentication
         is expected to occur only at another protocol layer.";
    }

    feature client-auth-x509-cert {
      description
        "Indicates that the server supports authenticating clients
         using X.509 certificates.";
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile";
    }

    feature client-auth-raw-public-key {
```

```
      description
        "Indicates that the server supports authenticating clients
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature client-auth-tls12-psk {
      description
        "Indicates that the server supports authenticating clients
         using PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature client-auth-tls13-epsk {
      description
        "Indicates that the server supports authenticating clients
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    // Groupings

    grouping tls-server-grouping {
      description
        "A reusable grouping for configuring a TLS server without
         any consideration for how underlying TCP sessions are
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tls-server-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container server-identity {
        nacm:default-deny-write;
        description
```

```
          "A locally-defined or referenced end-entity certificate,
           including any configured intermediate certificates, the
           TLS server will present when establishing a TLS connection
           in its Certificate message, as defined in Section 7.4.2
           in RFC 5246 and Section 4.4.2 in RFC 8446.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2
           RFC 8446: The Transport Layer Security (TLS) Protocol
                     Version 1.3
           RFC CCCC: A YANG Data Model for a Keystore";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst authentication types, of which one must
             be enabled (via its associated 'feature') and selected.";
          case certificate {
            if-feature "server-ident-x509-cert";
            container certificate {
              description
                "Specifies the server identity using a certificate.";
              uses
                ks:local-or-keystore-end-entity-cert-with-key-grouping{
                refine "local-or-keystore/local/local-definition" {
                  must 'public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
                }
                refine "local-or-keystore/keystore/keystore-reference"
                     + "/asymmetric-key" {
                  must 'deref(.)/../ks:public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
                }
              }
            }
          }
          case raw-private-key {
            if-feature "server-ident-raw-public-key";
            container raw-private-key {
              description
                "Specifies the server identity using a raw
                 private key.";
              uses ks:local-or-keystore-asymmetric-key-grouping {
                refine "local-or-keystore/local/local-definition" {
                  must 'public-key-format'
                     + ' = "ct:subject-public-key-info-format"';
                }
                refine "local-or-keystore/keystore/keystore-reference"{
                  must 'deref(.)/../ks:public-key-format'
```

```
                    + ' = "ct:subject-public-key-info-format"';
                }
              }
            }
          }
          case tls12-psk {
            if-feature "server-ident-tls12-psk";
            container tls12-psk {
              description
                "Specifies the server identity using a PSK (pre-shared
                 or pairwise-symmetric key).";
              uses ks:local-or-keystore-symmetric-key-grouping;
              leaf id_hint {
                type string;
                description
                  "The key 'psk_identity_hint' value used in the TLS
                   'ServerKeyExchange' message.";
                reference
                  "RFC 4279: Pre-Shared Key Ciphersuites for
                              Transport Layer Security (TLS)";
              }
            }
          }
          case tls13-epsk {
            if-feature "server-ident-tls13-epsk";
            container tls13-epsk {
              description
                "An External Pre-Shared Key (EPSK) is established
                or provisioned out-of-band, i.e., not from a TLS
                connection.  An EPSK is a tuple of (Base Key,
                External Identity, Hash).  External PSKs MUST
                NOT be imported for (D)TLS 1.2 or prior versions.
                When PSKs are provisioned out of band, the PSK
                identity and the KDF hash algorithm to be used
                with the PSK MUST also be provisioned.

                The structure of this container is designed
                to satisfy the requirements of RFC 8446
                Section 4.2.11, the recommendations from
                I-D ietf-tls-external-psk-guidance Section 6,
                and the EPSK input fields detailed in
                I-D draft-ietf-tls-external-psk-importer
                Section 3.1.  The base-key is based upon
                ks:local-or-keystore-symmetric-key-grouping
                in order to provide users with flexible and
                secure storage options.";
              reference
                "RFC 8446: The Transport Layer Security (TLS)
```

```
                    Protocol Version 1.3
                I-D.ietf-tls-external-psk-importer: Importing
                    External PSKs for TLS
                I-D.ietf-tls-external-psk-guidance: Guidance
                    for External PSK Usage in TLS";
              uses ks:local-or-keystore-symmetric-key-grouping;
              leaf external-identity {
                type string;
                mandatory true;
                description
                  "As per Section 4.2.11 of RFC 8446, and Section 4.1
                   of I-D. ietf-tls-external-psk-guidance: A sequence
                   of bytes used to identify an EPSK. A label for a
                   pre-shared key established externally.";
                reference
                  "RFC 8446: The Transport Layer Security (TLS)
                              Protocol Version 1.3
                   I-D.ietf-tls-external-psk-guidance:
                              Guidance for External PSK Usage in TLS";
              }
              leaf hash {
                type tlscmn:epsk-supported-hash;
                mandatory true;
                description
                  "As per Section 4.2.11 of RFC 8446, for externally
                   established PSKs, the Hash algorithm MUST be set
                   when the PSK is established or default to SHA-256
                   if no such algorithm is defined.  The server MUST
                   ensure that it selects a compatible PSK (if any)
                   and cipher suite.  Each PSK MUST only be used
                   with a single hash function.";
                reference
                  "RFC 8446: The Transport Layer Security (TLS)
                              Protocol Version 1.3";
              }
              leaf context {
                type string;
                description
                  "As per Section 4.1 of I-D.
                   ietf-tls-external-psk-guidance: Context
                   may include information about peer roles or
                   identities to mitigate Selfie-style reflection
                   attacks [Selfie].  If the EPSK is a key derived
                   from some other protocol or sequence of protocols,
                   context MUST include a channel binding for the
                   deriving protocols [RFC5056].  The details of
                   this binding are protocol specific.";
                reference
```

```
                  "I-D.ietf-tls-external-psk-importer:
                             Importing External PSKs for TLS
                   I-D.ietf-tls-external-psk-guidance:
                             Guidance for External PSK Usage in TLS";
              }
              leaf target-protocol {
                type uint16;
                description
                  "As per Section 3.1 of I-D.
                   ietf-tls-external-psk-guidance: The protocol
                   for which a PSK is imported for use.";
                reference
                  "I-D.ietf-tls-external-psk-importer:
                             Importing External PSKs for TLS";
              }
              leaf target-kdf {
                type uint16;
                description
                  "As per Section 3.1 of I-D.
                   ietf-tls-external-psk-guidance: The specific Key
                   Derivation Function (KDF) for which a PSK is
                   imported for use.";
                reference
                  "I-D.ietf-tls-external-psk-importer:
                             Importing External PSKs for TLS";
              }
            }
          }
        }
      } // container server-identity

      container client-authentication {
        if-feature "client-auth-supported";
        nacm:default-deny-write;
        must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
          or tls13-epsks';
        presence
          "Indicates that client authentication is supported (i.e.,
           that the server will request clients send certificates).
           If not configured, the TLS server SHOULD NOT request the
           TLS clients provide authentication credentials.";
        description
          "Specifies how the TLS server can authenticate TLS clients.
           Any combination of credentials is additive and unordered.

           Note that no configuration is required for PSK (pre-shared
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../server-
```

```
           identity' node.";
        container ca-certs {
          if-feature "client-auth-x509-cert";
          presence
            "Indicates that CA certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of certificate authority (CA) certificates used by
             the TLS server to authenticate TLS client certificates.
             A client certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "client-auth-x509-cert";
          presence
            "Indicates that EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of client certificates (i.e., end entity
             certificates) used by the TLS server to authenticate
             certificates presented by TLS clients. A client
             certificate is authenticated if it is an exact
             match to a configured client certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container raw-public-keys {
          if-feature "client-auth-raw-public-key";
          presence
            "Indicates that raw public keys have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of raw public keys used by the TLS server to
             authenticate raw public keys presented by the TLS
             client.  A raw public key is authenticated if it
             is an exact match to a configured raw public key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine "local-or-truststore/local/local-definition"
                 + "/public-key" {
```

```
              must 'public-key-format'
                 + ' = "ct:subject-public-key-info-format"';
            }
            refine "local-or-truststore/truststore"
                 + "/truststore-reference" {
              must 'deref(.)/../*/ts:public-key-format'
                 + ' = "ct:subject-public-key-info-format"';
            }
          }
        }
        leaf tls12-psks {
          if-feature "client-auth-tls12-psk";
          type empty;
          description
            "Indicates that the TLS server can authenticate TLS clients
             using configured PSKs (pre-shared or pairwise-symmetric
             keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'server-identity'
             node.";
        }
        leaf tls13-epsks {
          if-feature "client-auth-tls13-epsk";
          type empty;
          description
            "Indicates that the TLS 1.3 server can authenticate TLS
             clients using configured external PSKs (pre-shared keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'server-identity'
             node.";
        }
      } // container client-authentication

      container hello-params {
        nacm:default-deny-write;
        if-feature "tlscmn:hello-params";
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      } // container hello-params

      container keepalives {
        nacm:default-deny-write;
        if-feature "tls-server-keepalives";
        description
          "Configures the keepalive policy for the TLS server.";
```

```
          leaf peer-allowed-to-send {
            type empty;
            description
              "Indicates that the remote TLS client is allowed to send
               HeartbeatRequest messages, as defined by RFC 6520
               to this TLS server.";
            reference
              "RFC 6520: Transport Layer Security (TLS) and Datagram
               Transport Layer Security (DTLS) Heartbeat Extension";
          }
          container test-peer-aliveness {
            presence
              "Indicates that the TLS server proactively tests the
               aliveness of the remote TLS client.";
            description
              "Configures the keep-alive policy to proactively test
               the aliveness of the TLS client.  An unresponsive
               TLS client is dropped after approximately max-wait
               * max-attempts seconds.";
            leaf max-wait {
              type uint16 {
                range "1..max";
              }
              units "seconds";
              default "30";
              description
                "Sets the amount of time in seconds after which if
                 no data has been received from the TLS client, a
                 TLS-level message will be sent to test the
                 aliveness of the TLS client.";
            }
            leaf max-attempts {
              type uint8;
              default "3";
              description
                "Sets the maximum number of sequential keep-alive
                 messages that can fail to obtain a response from
                 the TLS client before assuming the TLS client is
                 no longer alive.";
            }
          }
        } // container keepalives
      } // grouping tls-server-grouping

   }

   <CODE ENDS>
```

5. Security Considerations

5.1. The "iana-tls-cipher-suite-algs" Module

The "iana-tls-cipher-suite-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. The "ietf-tls-common" YANG Module

The "ietf-tls-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.3. The "ietf-tls-client" YANG Module

The "ietf-tls-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.4. The "ietf-tls-server" YANG Module

The "ietf-tls-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers four URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs
Registrant Contact: IANA
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-common
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

6.2. The "YANG Module Names" Registry

This document registers four YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         iana-tls-cipher-suite-algs
namespace:    urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs
prefix:       tlscsa
reference:    RFC FFFF

name:         ietf-tls-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-common
prefix:       tlscmn
reference:    RFC FFFF

name:         ietf-tls-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-client
prefix:       tlsc
reference:    RFC FFFF

name:         ietf-tls-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-server
prefix:       tlss
reference:    RFC FFFF
```

6.3. The "iana-tls-cipher-suite-algs" Module

IANA is requested to maintain a YANG module called "iana-tls-cipher-suite-algs" that shadows the "TLS Cipher Suites" sub-registry of the "Transport Layer Security (TLS) Parameters" registry [IANA-CIPHER-ALGS].

This registry defines a YANG identity for each cipher suite algorithm, and a "base" identity from which all of the other identities are derived.

An initial version of this module can be found in Appendix A.1.

* Please note that this module was created on June 2st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

* Please also note that the "status" statement has been set to "deprecated", if the "RECOMMENDED" column in the registry had the value 'N', and to "obsolete", if the "References" column included Moving single-DES and IDEA TLS ciphersuites to Historic (https://datatracker.ietf.org/doc/status-change-tls-des-idea-ciphers-to-historic) reference.

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <https://www.rfc-editor.org/info/rfc5289>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[I-D.ietf-tls-external-psk-guidance]
Housley, R., Hoyland, J., Sethi, M., and C. A. Wood, "Guidance for External PSK Usage in TLS", Work in Progress, Internet-Draft, draft-ietf-tls-external-psk-guidance-06, 4 February 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-external-psk-guidance-06>.

[I-D.ietf-tls-external-psk-importer]
Benjamin, D. and C. A. Wood, "Importing External PSKs for TLS", Work in Progress, Internet-Draft, draft-ietf-tls-external-psk-importer-07, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-external-psk-importer-07>.

[IANA-CIPHER-ALGS]
(IANA), I. A. N. A., "IANA "TLS Cipher Suites" Sub-registry of the "Transport Layer Security (TLS) Parameters" Registry", <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4>.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <https://www.rfc-editor.org/info/rfc2246>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. YANG Modules for IANA

The module contained in this section was generated by scripts using the contents of the associated sub-registry as they existed on June 2nd, 2021.

A.1. Initial Module for the "TLS Cipher Suites" Registry

A.1.1. Data Model Overview

This section provides an overview of the "iana-tls-cipher-suite-algs" module in terms of its identities and protocol-accessible nodes.

A.1.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-CIPHER-ALGS].

```
Identities:
  +-- cipher-suite-alg-base
    +-- <identity-name from IANA registry>

    |  The diagram above uses syntax that is similar to but not
    |  defined in [RFC8340].
```

A.1.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-tls-cipher-suite-algs" module:

```
   Typedefs:
     identityref
       +-- cipher-suite-algorithm-ref

       | The diagram above uses syntax that is similar to but not
       | defined in [RFC8340].
```

Comments:

* The typedef defined in the "iana-tls-cipher-suite-algs" module extends the "identityref" type defined in [RFC7950].

A.1.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-tls-cipher-suite-alg" module:

```
   module: iana-tls-cipher-suite-algs
     +--ro supported-algorithms
        +--ro supported-algorithm*   cipher-suite-algorithm-ref
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

A.1.2. Example Usage

The following example illustrates operational state data indicating the TLS cipher suite algorithms supported by the server:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <supported-algorithms
     xmlns="urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs"
     xmlns:tlscsa="urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-al\
   gs">
     <supported-algorithm>tlscsa:tls-ecdhe-ecdsa-with-aes-256-cbc-sha</\
   supported-algorithm>
     <supported-algorithm>tlscsa:tls-dhe-rsa-with-aes-128-cbc-sha256</s\
   upported-algorithm>
     <supported-algorithm>tlscsa:tls-rsa-with-3des-ede-cbc-sha</support\
   ed-algorithm>
     <supported-algorithm>tlscsa:tls-ecdhe-psk-with-aes-256-gcm-sha384<\
   /supported-algorithm>
     <supported-algorithm>tlscsa:tls-dhe-psk-with-chacha20-poly1305-sha\
   256</supported-algorithm>
     <supported-algorithm>tlscsa:tls-eccpwd-with-aes-256-gcm-sha384</su\
   pported-algorithm>
     <supported-algorithm>tlscsa:tls-psk-with-aes-256-ccm</supported-al\
   gorithm>
     <supported-algorithm>tlscsa:tls-dhe-psk-with-camellia-256-cbc-sha3\
   84</supported-algorithm>
     <supported-algorithm>tlscsa:tls-ecdh-rsa-with-aes-256-cbc-sha384</\
   supported-algorithm>
     <supported-algorithm>tlscsa:tls-ecdh-rsa-with-3des-ede-cbc-sha</su\
   pported-algorithm>
     <supported-algorithm>tlscsa:tls-dh-dss-with-aes-128-gcm-sha256</su\
   pported-algorithm>
   </supported-algorithms>
```

A.1.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-02" reflects the day on which the extraction occurred.

```
   <CODE BEGINS> file "iana-tls-cipher-suite-algs@2021-06-02.yang"

   module iana-tls-cipher-suite-algs {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs";
     prefix tlscsa;

     organization
       "Internet Assigned Numbers Authority (IANA)";

     contact
       "Postal: ICANN
```

```
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA  90094-2536
               United States of America
          Tel:    +1 310 301 5800
          Email:  iana@iana.org";

      description
        "This module defines identities for the Cipher Suite
         algorithms defined in the 'TLS Cipher Suites' sub-registry
         of the 'Transport Layer Security (TLS) Parameters' registry
         maintained by IANA.

         Copyright (c) 2021 IETF Trust and the persons identified as
         authors of the code. All rights reserved.

         Redistribution and use in source and binary forms, with
         or without modification, is permitted pursuant to, and
         subject to the license terms contained in, the Revised
         BSD License set forth in Section 4.c of the IETF Trust's
         Legal Provisions Relating to IETF Documents
         (https://trustee.ietf.org/license-info).

         The initial version of this YANG module is part of RFC FFFF
         (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
         itself for full legal notices.";

      revision 2021-06-02 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Typedefs

      typedef cipher-suite-algorithm-ref {
        type identityref {
          base "cipher-suite-alg-base";
        }
        description
          "A reference to a TLS cipher suite algorithm identifier.";
      }


      // Identities

      identity cipher-suite-alg-base {
        description
```

```
        "Base identity used to identify TLS cipher suites.";
    }

    identity tls-null-with-null-null {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-NULL-WITH-NULL-NULL";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-with-null-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-NULL-MD5";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-with-null-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-NULL-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-export-with-rc4-40-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-EXPORT-WITH-RC4-40-MD5";
      reference
        "RFC 4346:
           The TLS Protocol Version 1.1
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-rsa-with-rc4-128-md5 {
      base cipher-suite-alg-base;
      status deprecated;
```

```
      description
        "TLS-RSA-WITH-RC4-128-MD5";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-rsa-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-RC4-128-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-rsa-export-with-rc2-cbc-40-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-EXPORT-WITH-RC2-CBC-40-MD5";
      reference
        "RFC 4346:
           The TLS Protocol Version 1.1";
    }

    identity tls-rsa-with-idea-cbc-sha {
      base cipher-suite-alg-base;
      status obsolete;
      description
        "TLS-RSA-WITH-IDEA-CBC-SHA";
      reference
        "RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)
         RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-export-with-des40-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
```

```
      description
        "TLS-RSA-EXPORT-WITH-DES40-CBC-SHA";
      reference
        "RFC 4346:
           The TLS Protocol Version 1.1";
    }

    identity tls-rsa-with-des-cbc-sha {
      base cipher-suite-alg-base;
      status obsolete;
      description
        "TLS-RSA-WITH-DES-CBC-SHA";
      reference
        "RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)
         RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-dss-export-with-des40-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-EXPORT-WITH-DES40-CBC-SHA";
      reference
        "RFC 4346:
           The TLS Protocol Version 1.1";
    }

    identity tls-dh-dss-with-des-cbc-sha {
      base cipher-suite-alg-base;
      status obsolete;
      description
        "TLS-DH-DSS-WITH-DES-CBC-SHA";
      reference
        "RFC 5469:
```

```
            DES and IDEA Cipher Suites for
            Transport Layer Security (TLS)
          RFC 5469:
            DES and IDEA Cipher Suites for
            Transport Layer Security (TLS)";
      }

      identity tls-dh-dss-with-3des-ede-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DH-DSS-WITH-3DES-EDE-CBC-SHA";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version 1.2";
      }

      identity tls-dh-rsa-export-with-des40-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DH-RSA-EXPORT-WITH-DES40-CBC-SHA";
        reference
          "RFC 4346:
             The TLS Protocol Version 1.1";
      }

      identity tls-dh-rsa-with-des-cbc-sha {
        base cipher-suite-alg-base;
        status obsolete;
        description
          "TLS-DH-RSA-WITH-DES-CBC-SHA";
        reference
          "RFC 5469:
             DES and IDEA Cipher Suites for
             Transport Layer Security (TLS)
           RFC 5469:
             DES and IDEA Cipher Suites for
             Transport Layer Security (TLS)";
      }

      identity tls-dh-rsa-with-3des-ede-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DH-RSA-WITH-3DES-EDE-CBC-SHA";
        reference
          "RFC 5246:
```

```
          The Transport Layer Security (TLS) Protocol Version 1.2";
      }

      identity tls-dhe-dss-export-with-des40-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DHE-DSS-EXPORT-WITH-DES40-CBC-SHA";
        reference
          "RFC 4346:
             The TLS Protocol Version 1.1";
      }

      identity tls-dhe-dss-with-des-cbc-sha {
        base cipher-suite-alg-base;
        status obsolete;
        description
          "TLS-DHE-DSS-WITH-DES-CBC-SHA";
        reference
          "RFC 5469:
             DES and IDEA Cipher Suites for
             Transport Layer Security (TLS)
           RFC 5469:
             DES and IDEA Cipher Suites for
             Transport Layer Security (TLS)";
      }

      identity tls-dhe-dss-with-3des-ede-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DHE-DSS-WITH-3DES-EDE-CBC-SHA";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version 1.2";
      }

      identity tls-dhe-rsa-export-with-des40-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DHE-RSA-EXPORT-WITH-DES40-CBC-SHA";
        reference
          "RFC 4346:
             The TLS Protocol Version 1.1";
      }

      identity tls-dhe-rsa-with-des-cbc-sha {
```

```
      base cipher-suite-alg-base;
      status obsolete;
      description
        "TLS-DHE-RSA-WITH-DES-CBC-SHA";
      reference
        "RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)
         RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-anon-export-with-rc4-40-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-EXPORT-WITH-RC4-40-MD5";
      reference
        "RFC 4346:
           The TLS Protocol Version 1.1
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-dh-anon-with-rc4-128-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-RC4-128-MD5";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-dh-anon-export-with-des40-cbc-sha {
```

```
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-EXPORT-WITH-DES40-CBC-SHA";
      reference
        "RFC 4346:
           The TLS Protocol Version 1.1";
    }

    identity tls-dh-anon-with-des-cbc-sha {
      base cipher-suite-alg-base;
      status obsolete;
      description
        "TLS-DH-ANON-WITH-DES-CBC-SHA";
      reference
        "RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)
         RFC 5469:
           DES and IDEA Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-krb5-with-des-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-WITH-DES-CBC-SHA";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-krb5-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-KRB5-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-krb5-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-WITH-RC4-128-SHA";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-krb5-with-idea-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-WITH-IDEA-CBC-SHA";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-krb5-with-des-cbc-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-WITH-DES-CBC-MD5";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-krb5-with-3des-ede-cbc-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-WITH-3DES-EDE-CBC-MD5";
      reference
```

```
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-krb5-with-rc4-128-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-WITH-RC4-128-MD5";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-krb5-with-idea-cbc-md5 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-WITH-IDEA-CBC-MD5";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-krb5-export-with-des-cbc-40-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-EXPORT-WITH-DES-CBC-40-SHA";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-krb5-export-with-rc2-cbc-40-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-KRB5-EXPORT-WITH-RC2-CBC-40-SHA";
      reference
        "RFC 2712:
           Addition of Kerberos Cipher Suites to
```

```
          Transport Layer Security (TLS)";
      }

      identity tls-krb5-export-with-rc4-40-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-KRB5-EXPORT-WITH-RC4-40-SHA";
        reference
          "RFC 2712:
             Addition of Kerberos Cipher Suites to
             Transport Layer Security (TLS)
           RFC 6347:
             Datagram Transport Layer Security version 1.2";
      }

      identity tls-krb5-export-with-des-cbc-40-md5 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-KRB5-EXPORT-WITH-DES-CBC-40-MD5";
        reference
          "RFC 2712:
             Addition of Kerberos Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-krb5-export-with-rc2-cbc-40-md5 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-KRB5-EXPORT-WITH-RC2-CBC-40-MD5";
        reference
          "RFC 2712:
             Addition of Kerberos Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-krb5-export-with-rc4-40-md5 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-KRB5-EXPORT-WITH-RC4-40-MD5";
        reference
          "RFC 2712:
             Addition of Kerberos Cipher Suites to
             Transport Layer Security (TLS)
           RFC 6347:
```

```
          Datagram Transport Layer Security version 1.2";
      }

      identity tls-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-PSK-WITH-NULL-SHA";
        reference
          "RFC 4785:
             Pre-Shared Key Cipher Suites with NULL Encryption for
             Transport Layer Security (TLS)";
      }

      identity tls-dhe-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DHE-PSK-WITH-NULL-SHA";
        reference
          "RFC 4785:
             Pre-Shared Key Cipher Suites with NULL Encryption for
             Transport Layer Security (TLS)";
      }

      identity tls-rsa-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-RSA-PSK-WITH-NULL-SHA";
        reference
          "RFC 4785:
             Pre-Shared Key Cipher Suites with NULL Encryption for
             Transport Layer Security (TLS)";
      }

      identity tls-rsa-with-aes-128-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-RSA-WITH-AES-128-CBC-SHA";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version 1.2";
      }

      identity tls-dh-dss-with-aes-128-cbc-sha {
        base cipher-suite-alg-base;
```

```
      status deprecated;
      description
        "TLS-DH-DSS-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-rsa-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-dss-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-rsa-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-anon-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }
```

```
    identity tls-rsa-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-dss-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-rsa-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-dss-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-rsa-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
```

```
    }

    identity tls-dh-anon-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-with-null-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-NULL-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-with-aes-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-256-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-dss-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-AES-128-CBC-SHA256";
      reference
```

```
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-rsa-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-dss-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-with-camellia-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-CAMELLIA-128-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-dss-with-camellia-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-CAMELLIA-128-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-rsa-with-camellia-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-DH-RSA-WITH-CAMELLIA-128-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-dss-with-camellia-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-CAMELLIA-128-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-camellia-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-CAMELLIA-128-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-anon-with-camellia-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-CAMELLIA-128-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-dss-with-aes-256-cbc-sha256 {
      base cipher-suite-alg-base;
```

```
      status deprecated;
      description
        "TLS-DH-DSS-WITH-AES-256-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-rsa-with-aes-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-AES-256-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-dss-with-aes-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-AES-256-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-rsa-with-aes-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-AES-256-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-anon-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }
```

```
    identity tls-dh-anon-with-aes-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-AES-256-CBC-SHA256";
      reference
        "RFC 5246:
           The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-rsa-with-camellia-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-CAMELLIA-256-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-dss-with-camellia-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-CAMELLIA-256-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-rsa-with-camellia-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-CAMELLIA-256-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-dss-with-camellia-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-CAMELLIA-256-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
```

```
    }

    identity tls-dhe-rsa-with-camellia-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-CAMELLIA-256-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-anon-with-camellia-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-CAMELLIA-256-CBC-SHA";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-psk-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-RC4-128-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-psk-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
```

```
      status deprecated;
      description
        "TLS-PSK-WITH-AES-128-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-256-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-RC4-128-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-dhe-psk-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-DHE-PSK-WITH-AES-128-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-AES-256-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-RC4-128-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-rsa-psk-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-AES-128-CBC-SHA";
      reference
```

```
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-AES-256-CBC-SHA";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-seed-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-SEED-CBC-SHA";
      reference
        "RFC 4162:
           Addition of SEED Ciphersuites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-dss-with-seed-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-SEED-CBC-SHA";
      reference
        "RFC 4162:
           Addition of SEED Ciphersuites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-rsa-with-seed-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-SEED-CBC-SHA";
      reference
        "RFC 4162:
           Addition of SEED Ciphersuites to
           Transport Layer Security (TLS)";
    }
```

```
    identity tls-dhe-dss-with-seed-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-SEED-CBC-SHA";
      reference
        "RFC 4162:
           Addition of SEED Ciphersuites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-seed-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-SEED-CBC-SHA";
      reference
        "RFC 4162:
           Addition of SEED Ciphersuites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-seed-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-SEED-CBC-SHA";
      reference
        "RFC 4162:
           Addition of SEED Ciphersuites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-rsa-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-256-GCM-SHA384";
```

```
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-RSA-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-RSA-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dh-rsa-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dh-rsa-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dhe-dss-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-AES-128-GCM-SHA256";
```

```
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dhe-dss-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dh-dss-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dh-dss-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dh-anon-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-dh-anon-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
```

```
      description
        "TLS-DH-ANON-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5288:
           AES-GCM Cipher Suites for TLS";
    }

    identity tls-psk-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-psk-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-dhe-psk-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-PSK-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-dhe-psk-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-PSK-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }
```

```
    identity tls-rsa-psk-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-psk-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-psk-with-aes-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-256-CBC-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-psk-with-null-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-PSK-WITH-NULL-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-psk-with-null-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-NULL-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-dhe-psk-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-dhe-psk-with-aes-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-AES-256-CBC-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-dhe-psk-with-null-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-NULL-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
```

```
          (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-dhe-psk-with-null-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-NULL-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-aes-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-AES-256-CBC-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-null-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-NULL-SHA256";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-null-sha384 {
```

```
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-NULL-SHA384";
      reference
        "RFC 5487:
           Pre-Shared Key Cipher Suites for Transport Layer Security
           (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-dss-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-rsa-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-dss-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
```

```
    }

    identity tls-dhe-rsa-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-anon-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-rsa-with-camellia-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-CAMELLIA-256-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-dss-with-camellia-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-CAMELLIA-256-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-rsa-with-camellia-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-CAMELLIA-256-CBC-SHA256";
      reference
```

```
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-dss-with-camellia-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-CAMELLIA-256-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-camellia-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-CAMELLIA-256-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-dh-anon-with-camellia-256-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-CAMELLIA-256-CBC-SHA256";
      reference
        "RFC 5932:
           Camellia Cipher Suites for TLS";
    }

    identity tls-sm4-gcm-sm3 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-SM4-GCM-SM3";
      reference
        "RFC 8998:
           ShangMi (SM) Cipher Suites for Transport Layer Security
           (TLS) Protocol Version 1.3";
    }

    identity tls-sm4-ccm-sm3 {
      base cipher-suite-alg-base;
      status deprecated;
```

```
      description
        "TLS-SM4-CCM-SM3";
      reference
        "RFC 8998:
           ShangMi (SM) Cipher Suites for Transport Layer Security
           (TLS) Protocol Version 1.3";
    }

    identity tls-empty-renegotiation-info-scsv {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-EMPTY-RENEGOTIATION-INFO-SCSV";
      reference
        "RFC 5746:
           Transport Layer Security (TLS)
           Renegotiation Indication Extension";
    }

    identity tls-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-AES-128-GCM-SHA256";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity tls-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      description
        "TLS-AES-256-GCM-SHA384";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity tls-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-CHACHA20-POLY1305-SHA256";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity tls-aes-128-ccm-sha256 {
      base cipher-suite-alg-base;
```

```
      description
        "TLS-AES-128-CCM-SHA256";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity tls-aes-128-ccm-8-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-AES-128-CCM-8-SHA256";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity tls-fallback-scsv {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-FALLBACK-SCSV";
      reference
        "RFC 7507:
           TLS Fallback Signaling Cipher Suite Value (SCSV)
           for Preventing Protocol Downgrade Attacks";
    }

    identity tls-ecdh-ecdsa-with-null-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-NULL-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-ecdsa-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-RC4-128-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier
```

```
          RFC 6347:
            Datagram Transport Layer Security version 1.2";
      }

      identity tls-ecdh-ecdsa-with-3des-ede-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDH-ECDSA-WITH-3DES-EDE-CBC-SHA";
        reference
          "RFC 8422:
             Elliptic Curve Cryptography (ECC) Cipher Suites for
             Transport Layer Security (TLS) Versions 1.2 and Earlier";
      }

      identity tls-ecdh-ecdsa-with-aes-128-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDH-ECDSA-WITH-AES-128-CBC-SHA";
        reference
          "RFC 8422:
             Elliptic Curve Cryptography (ECC) Cipher Suites for
             Transport Layer Security (TLS) Versions 1.2 and Earlier";
      }

      identity tls-ecdh-ecdsa-with-aes-256-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDH-ECDSA-WITH-AES-256-CBC-SHA";
        reference
          "RFC 8422:
             Elliptic Curve Cryptography (ECC) Cipher Suites for
             Transport Layer Security (TLS) Versions 1.2 and Earlier";
      }

      identity tls-ecdhe-ecdsa-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDHE-ECDSA-WITH-NULL-SHA";
        reference
          "RFC 8422:
             Elliptic Curve Cryptography (ECC) Cipher Suites for
             Transport Layer Security (TLS) Versions 1.2 and Earlier";
      }
```

```
    identity tls-ecdhe-ecdsa-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-RC4-128-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-ecdhe-ecdsa-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdhe-ecdsa-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-128-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdhe-ecdsa-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-rsa-with-null-sha {
      base cipher-suite-alg-base;
```

```
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-NULL-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-rsa-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-RC4-128-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-ecdh-rsa-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-rsa-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-AES-128-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-rsa-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-ECDH-RSA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdhe-rsa-with-null-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-NULL-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdhe-rsa-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-RC4-128-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-ecdhe-rsa-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdhe-rsa-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA";
      reference
```

```
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdhe-rsa-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-anon-with-null-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ANON-WITH-NULL-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-anon-with-rc4-128-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ANON-WITH-RC4-128-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-ecdh-anon-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ANON-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
```

```
          Transport Layer Security (TLS) Versions 1.2 and Earlier";
      }

      identity tls-ecdh-anon-with-aes-128-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDH-ANON-WITH-AES-128-CBC-SHA";
        reference
          "RFC 8422:
             Elliptic Curve Cryptography (ECC) Cipher Suites for
             Transport Layer Security (TLS) Versions 1.2 and Earlier";
      }

      identity tls-ecdh-anon-with-aes-256-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDH-ANON-WITH-AES-256-CBC-SHA";
        reference
          "RFC 8422:
             Elliptic Curve Cryptography (ECC) Cipher Suites for
             Transport Layer Security (TLS) Versions 1.2 and Earlier";
      }

      identity tls-srp-sha-with-3des-ede-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-SRP-SHA-WITH-3DES-EDE-CBC-SHA";
        reference
          "RFC 5054:
             Using SRP for TLS Authentication";
      }

      identity tls-srp-sha-rsa-with-3des-ede-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-SRP-SHA-RSA-WITH-3DES-EDE-CBC-SHA";
        reference
          "RFC 5054:
             Using SRP for TLS Authentication";
      }

      identity tls-srp-sha-dss-with-3des-ede-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
```

```
      description
        "TLS-SRP-SHA-DSS-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 5054:
           Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-SRP-SHA-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5054:
           Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-rsa-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-SRP-SHA-RSA-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5054:
           Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-dss-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-SRP-SHA-DSS-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5054:
           Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-SRP-SHA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5054:
           Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-rsa-with-aes-256-cbc-sha {
```

```
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-SRP-SHA-RSA-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5054:
           Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-dss-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-SRP-SHA-DSS-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5054:
           Using SRP for TLS Authentication";
    }

    identity tls-ecdhe-ecdsa-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdhe-ecdsa-with-aes-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-256-CBC-SHA384";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdh-ecdsa-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5289:
```

```
            TLS Elliptic Curve Cipher Suites with SHA-256/384
            and AES Galois Counter Mode";
      }

      identity tls-ecdh-ecdsa-with-aes-256-cbc-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDH-ECDSA-WITH-AES-256-CBC-SHA384";
        reference
          "RFC 5289:
             TLS Elliptic Curve Cipher Suites with SHA-256/384
             and AES Galois Counter Mode";
      }

      identity tls-ecdhe-rsa-with-aes-128-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA256";
        reference
          "RFC 5289:
             TLS Elliptic Curve Cipher Suites with SHA-256/384
             and AES Galois Counter Mode";
      }

      identity tls-ecdhe-rsa-with-aes-256-cbc-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDHE-RSA-WITH-AES-256-CBC-SHA384";
        reference
          "RFC 5289:
             TLS Elliptic Curve Cipher Suites with SHA-256/384
             and AES Galois Counter Mode";
      }

      identity tls-ecdh-rsa-with-aes-128-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDH-RSA-WITH-AES-128-CBC-SHA256";
        reference
          "RFC 5289:
             TLS Elliptic Curve Cipher Suites with SHA-256/384
             and AES Galois Counter Mode";
      }
```

```
    identity tls-ecdh-rsa-with-aes-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-AES-256-CBC-SHA384";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdhe-ecdsa-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdhe-ecdsa-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdh-ecdsa-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdh-ecdsa-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-AES-256-GCM-SHA384";
      reference
```

```
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdhe-rsa-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdhe-rsa-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdh-rsa-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdh-rsa-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 5289:
           TLS Elliptic Curve Cipher Suites with SHA-256/384
           and AES Galois Counter Mode";
    }

    identity tls-ecdhe-psk-with-rc4-128-sha {
```

```
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-RC4-128-SHA";
      reference
        "RFC 5489:
           ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)
         RFC 6347:
           Datagram Transport Layer Security version 1.2";
    }

    identity tls-ecdhe-psk-with-3des-ede-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-3DES-EDE-CBC-SHA";
      reference
        "RFC 5489:
           ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-aes-128-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA";
      reference
        "RFC 5489:
           ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-aes-256-cbc-sha {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-AES-256-CBC-SHA";
      reference
        "RFC 5489:
           ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-aes-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256";
      reference
        "RFC 5489:
```

```
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
      }

      identity tls-ecdhe-psk-with-aes-256-cbc-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDHE-PSK-WITH-AES-256-CBC-SHA384";
        reference
          "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
      }

      identity tls-ecdhe-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDHE-PSK-WITH-NULL-SHA";
        reference
          "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
      }

      identity tls-ecdhe-psk-with-null-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDHE-PSK-WITH-NULL-SHA256";
        reference
          "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
      }

      identity tls-ecdhe-psk-with-null-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-ECDHE-PSK-WITH-NULL-SHA384";
        reference
          "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
      }

      identity tls-rsa-with-aria-128-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-RSA-WITH-ARIA-128-CBC-SHA256";
```

```
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-dss-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-dss-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-rsa-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
```

```
    }

    identity tls-dh-rsa-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-dss-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-dss-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
```

```
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-ecdsa-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-ecdsa-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-ARIA-256-CBC-SHA384";
      reference
```

```
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }
```

```
    identity tls-ecdh-rsa-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-rsa-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-DHE-RSA-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-rsa-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-rsa-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-dss-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
```

```
          Transport Layer Security (TLS)";
      }

      identity tls-dhe-dss-with-aria-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DHE-DSS-WITH-ARIA-256-GCM-SHA384";
        reference
          "RFC 6209:
             Addition of the ARIA Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-dh-dss-with-aria-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DH-DSS-WITH-ARIA-128-GCM-SHA256";
        reference
          "RFC 6209:
             Addition of the ARIA Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-dh-dss-with-aria-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DH-DSS-WITH-ARIA-256-GCM-SHA384";
        reference
          "RFC 6209:
             Addition of the ARIA Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-dh-anon-with-aria-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DH-ANON-WITH-ARIA-128-GCM-SHA256";
        reference
          "RFC 6209:
             Addition of the ARIA Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-dh-anon-with-aria-256-gcm-sha384 {
```

```
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-ecdsa-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-ecdsa-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-ARIA-256-GCM-SHA384";
```

```
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-rsa-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-rsa-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
```

```
  }

  identity tls-psk-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-PSK-WITH-ARIA-128-CBC-SHA256";
    reference
      "RFC 6209:
         Addition of the ARIA Cipher Suites to
         Transport Layer Security (TLS)";
  }

  identity tls-psk-with-aria-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-PSK-WITH-ARIA-256-CBC-SHA384";
    reference
      "RFC 6209:
         Addition of the ARIA Cipher Suites to
         Transport Layer Security (TLS)";
  }

  identity tls-dhe-psk-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DHE-PSK-WITH-ARIA-128-CBC-SHA256";
    reference
      "RFC 6209:
         Addition of the ARIA Cipher Suites to
         Transport Layer Security (TLS)";
  }

  identity tls-dhe-psk-with-aria-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DHE-PSK-WITH-ARIA-256-CBC-SHA384";
    reference
      "RFC 6209:
         Addition of the ARIA Cipher Suites to
         Transport Layer Security (TLS)";
  }

  identity tls-rsa-psk-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
```

```
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-ARIA-128-GCM-SHA256";
      reference
```

```
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-aria-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-ARIA-128-GCM-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-aria-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-ARIA-256-GCM-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-aria-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-ARIA-128-CBC-SHA256";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }
```

```
    identity tls-ecdhe-psk-with-aria-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-ARIA-256-CBC-SHA384";
      reference
        "RFC 6209:
           Addition of the ARIA Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-ecdsa-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-ecdsa-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-ECDH-ECDSA-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-rsa-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-rsa-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
```

```
          Transport Layer Security (TLS)";
      }

      identity tls-rsa-with-camellia-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-RSA-WITH-CAMELLIA-128-GCM-SHA256";
        reference
          "RFC 6367:
             Addition of the Camellia Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-rsa-with-camellia-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-RSA-WITH-CAMELLIA-256-GCM-SHA384";
        reference
          "RFC 6367:
             Addition of the Camellia Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-dhe-rsa-with-camellia-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DHE-RSA-WITH-CAMELLIA-128-GCM-SHA256";
        reference
          "RFC 6367:
             Addition of the Camellia Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-dhe-rsa-with-camellia-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
          "TLS-DHE-RSA-WITH-CAMELLIA-256-GCM-SHA384";
        reference
          "RFC 6367:
             Addition of the Camellia Cipher Suites to
             Transport Layer Security (TLS)";
      }

      identity tls-dh-rsa-with-camellia-128-gcm-sha256 {
```

```
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-rsa-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-RSA-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-dss-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-dss-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-DSS-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-dss-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-CAMELLIA-128-GCM-SHA256";
```

```
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-dss-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-DSS-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DH-ANON-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-ecdsa-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
```

```
    }

    identity tls-ecdhe-ecdsa-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-ecdsa-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-ECDSA-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
```

```
      status deprecated;
      description
        "TLS-ECDHE-RSA-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-rsa-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdh-rsa-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDH-RSA-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-CAMELLIA-256-GCM-SHA384";
      reference
```

```
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-camellia-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-CAMELLIA-128-GCM-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-camellia-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-CAMELLIA-256-GCM-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }
```

```
    identity tls-psk-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-PSK-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-RSA-PSK-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-camellia-128-cbc-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-CAMELLIA-128-CBC-SHA256";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-camellia-256-cbc-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-CAMELLIA-256-CBC-SHA384";
      reference
        "RFC 6367:
           Addition of the Camellia Cipher Suites to
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-aes-128-ccm {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-128-CCM";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
```

```
    }

    identity tls-rsa-with-aes-256-ccm {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-256-CCM";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-aes-128-ccm {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-RSA-WITH-AES-128-CCM";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-aes-256-ccm {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-RSA-WITH-AES-256-CCM";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-rsa-with-aes-128-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-128-CCM-8";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-rsa-with-aes-256-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-WITH-AES-256-CCM-8";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
```

```
    }

    identity tls-dhe-rsa-with-aes-128-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-AES-128-CCM-8";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-aes-256-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-DHE-RSA-WITH-AES-256-CCM-8";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-with-aes-128-ccm {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-128-CCM";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-with-aes-256-ccm {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-256-CCM";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-dhe-psk-with-aes-128-ccm {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-PSK-WITH-AES-128-CCM";
      reference
        "RFC 6655:
```

```
          AES-CCM Cipher Suites for TLS";
    }

    identity tls-dhe-psk-with-aes-256-ccm {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-PSK-WITH-AES-256-CCM";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-with-aes-128-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-128-CCM-8";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-with-aes-256-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-AES-256-CCM-8";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-dhe-with-aes-128-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-DHE-WITH-AES-128-CCM-8";
      reference
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-dhe-with-aes-256-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-DHE-WITH-AES-256-CCM-8";
      reference
```

```
        "RFC 6655:
           AES-CCM Cipher Suites for TLS";
    }

    identity tls-ecdhe-ecdsa-with-aes-128-ccm {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-128-CCM";
      reference
        "RFC 7251:
           AES-CCM ECC Cipher Suites for TLS";
    }

    identity tls-ecdhe-ecdsa-with-aes-256-ccm {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-256-CCM";
      reference
        "RFC 7251:
           AES-CCM ECC Cipher Suites for TLS";
    }

    identity tls-ecdhe-ecdsa-with-aes-128-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-128-CCM-8";
      reference
        "RFC 7251:
           AES-CCM ECC Cipher Suites for TLS";
    }

    identity tls-ecdhe-ecdsa-with-aes-256-ccm-8 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-ECDSA-WITH-AES-256-CCM-8";
      reference
        "RFC 7251:
           AES-CCM ECC Cipher Suites for TLS";
    }

    identity tls-eccpwd-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
```

```
        "TLS-ECCPWD-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 8492:
           Secure Password Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-eccpwd-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECCPWD-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 8492:
           Secure Password Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-eccpwd-with-aes-128-ccm-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECCPWD-WITH-AES-128-CCM-SHA256";
      reference
        "RFC 8492:
           Secure Password Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-eccpwd-with-aes-256-ccm-sha384 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECCPWD-WITH-AES-256-CCM-SHA384";
      reference
        "RFC 8492:
           Secure Password Ciphersuites for
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-rsa-with-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-RSA-WITH-CHACHA20-POLY1305-SHA256";
      reference
        "RFC 7905:
           ChaCha20-Poly1305 Cipher Suites for
           Transport Layer Security (TLS)";
```

```
    }

    identity tls-ecdhe-ecdsa-with-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-ECDSA-WITH-CHACHA20-POLY1305-SHA256";
      reference
        "RFC 7905:
           ChaCha20-Poly1305 Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-RSA-WITH-CHACHA20-POLY1305-SHA256";
      reference
        "RFC 7905:
           ChaCha20-Poly1305 Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-psk-with-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-PSK-WITH-CHACHA20-POLY1305-SHA256";
      reference
        "RFC 7905:
           ChaCha20-Poly1305 Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-PSK-WITH-CHACHA20-POLY1305-SHA256";
      reference
        "RFC 7905:
           ChaCha20-Poly1305 Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-DHE-PSK-WITH-CHACHA20-POLY1305-SHA256";
      reference
```

```
        "RFC 7905:
           ChaCha20-Poly1305 Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-chacha20-poly1305-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-RSA-PSK-WITH-CHACHA20-POLY1305-SHA256";
      reference
        "RFC 7905:
           ChaCha20-Poly1305 Cipher Suites for
           Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-aes-128-gcm-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-PSK-WITH-AES-128-GCM-SHA256";
      reference
        "RFC 8442:
           ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
    }

    identity tls-ecdhe-psk-with-aes-256-gcm-sha384 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-PSK-WITH-AES-256-GCM-SHA384";
      reference
        "RFC 8442:
           ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
    }

    identity tls-ecdhe-psk-with-aes-128-ccm-8-sha256 {
      base cipher-suite-alg-base;
      status deprecated;
      description
        "TLS-ECDHE-PSK-WITH-AES-128-CCM-8-SHA256";
      reference
        "RFC 8442:
           ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
    }

    identity tls-ecdhe-psk-with-aes-128-ccm-sha256 {
      base cipher-suite-alg-base;
      description
        "TLS-ECDHE-PSK-WITH-AES-128-CCM-SHA256";
```

```
        reference
          "RFC 8442:
             ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
      }

      // Protocol-accessible Nodes

      container supported-algorithms {
        config false;
        description
          "A container for a list of cipher suite algorithms supported
           by the server.";
        leaf-list supported-algorithm {
          type cipher-suite-algorithm-ref;
          description
            "A cipher suite algorithm supported by the server.";
        }
      }

    }

    <CODE ENDS>
```

Appendix B. Change Log

This section is to be removed before publishing as an RFC.

B.1. 00 to 01

* Noted that '0.0.0.0' and '::' might have special meanings.

* Renamed "keychain" to "keystore".

B.2. 01 to 02

* Removed the groupings containing transport-level configuration. Now modules contain only the transport-independent groupings.

* Filled in previously incomplete 'ietf-tls-client' module.

* Added cipher suites for various algorithms into new 'ietf-tls-common' module.

B.3. 02 to 03

* Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

* Fixed description statement for leaf 'trusted-ca-certs'.

B.4. 03 to 04

* Updated title to "YANG Groupings for TLS Clients and TLS Servers"
* Updated leafref paths to point to new keystore path
* Changed the YANG prefix for ietf-tls-common from 'tlscom' to 'tlscmn'.
* Added TLS protocol verions 1.0 and 1.1.
* Made author lists consistent
* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Updated YANG to use typedefs around leafrefs to common keystore paths
* Now inlines key and certificates (no longer a leafref to keystore)

B.5. 04 to 05

* Merged changes from co-author.

B.6. 05 to 06

* Updated to use trust anchors from trust-anchors draft (was keystore draft)
* Now Uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

B.7. 06 to 07

* factored the tls-[client|server]-groupings into more reusable groupings.
* added if-feature statements for the new "x509-certificates" feature defined in draft-ietf-netconf-trust-anchors.

B.8. 07 to 08

* Added a number of compatibility matrices to Section 5 (thanks Frank!)

* Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

B.9. 08 to 09

* Updated examples to reflect update to groupings defined in the keystore draft.

* Add TLS keepalives features and groupings.

* Prefixed top-level TLS grouping nodes with 'tls-' and support mashups.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

* Reformatted the YANG modules.

B.11. 10 to 11

* Collapsed all the inner groupings into the top-level grouping.

* Added a top-level "demux container" inside the top-level grouping.

* Added NACM statements and updated the Security Considerations section.

* Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

B.12. 11 to 12

* In server model, made 'client-authentication' a 'presence' node indicating that the server supports client authentication.

* In the server model, added a 'required-or-optional' choice to 'client-authentication' to better support protocols such as RESTCONF.

* In the server model, added a 'local-or-external' choice to 'client-authentication' to better support consuming data models that prefer to keep client auth with client definitions than in a model principally concerned with the "transport".

* In both models, removed the "demux containers", floating the nacm:default-deny-write to each descendant node, and adding a note to model designers regarding the potential need to add their own demux containers.

* Fixed a couple references (section 2 --> section 3)

B.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

B.14. 12 to 13

* Removed 'container' under 'client-identity' to match server model.

* Updated examples to reflect change grouping in keystore module.

B.15. 13 to 14

* Removed the "certificate" container from "client-identity" in the ietf-tls-client module.

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

B.16. 14 to 15

* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:certificates-ref" to a container that uses "ts:local-or-truststore-certs-grouping".

B.17. 15 to 16

* Removed unnecessary if-feature statements in the -client and -server modules.

* Cleaned up some description statements in the -client and -server modules.

* Fixed a canonical ordering issue in ietf-tls-common detected by new pyang.

B.18. 16 to 17

* Removed choice local-or-external by removing the 'external' case and flattening the 'local' case and adding a "client-auth-supported" feature.

* Removed choice required-or-optional.

* Updated examples to include the "*-key-format" nodes.

* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:tls-public-key-format" (must expr for ref'ed keys are TBD).

B.19. 17 to 18

* Removed the unused "external-client-auth-supported" feature.

* Made client-indentity optional, as there may be over-the-top auth instead.

* Added augment to uses of local-or-keystore-symmetric-key-grouping for a psk "id" node.

* Added missing presence container "psks" to ietf-tls-server's "client-authentication" container.

* Updated examples to reflect new "bag" addition to truststore.

* Removed feature-limited caseless 'case' statements to improve tree diagram rendering.

* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.

* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags).

* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

B.20. 18 to 19

* Updated the "keepalives" containers in part to address Michal Vasko's request to align with RFC 8071, and in part to better align to RFC 6520.

* Removed algorithm-mapping tables from the "TLS Common Model" section

* Removed the 'algorithm' node from the examples.

* Renamed both "client-certs" and "server-certs" to "ee-certs"

* Added a "Note to Reviewers" note to first page.

B.21. 19 to 20

* Modified the 'must' expression in the "ietf-tls-client:server-authention" node to cover the "raw-public-keys" and "psks" nodes also.

* Added a "must 'ca-certs or ee-certs or raw-public-keys or psks'" statement to the ietf-tls-server:client-authentication" node.

* Added "mandatory true" to "choice auth-type" and a "presence" statement to its ancestor.

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Moved the "ietf-tls-common" module section to proceed the other two module sections.

* Updated the Security Considerations section.

B.22. 20 to 21

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

B.23. 21 to 22

* In both the "client-authentication" and "server-authentication" subtrees, replaced the "psks" node from being a P-container to a leaf of type "empty".

* Cleaned up examples (e.g., removed FIXMEs)

* Fixed issues found by the SecDir review of the "keystore" draft.

* Updated the "psk" sections in the "ietf-tls-client" and "ietf-tls-server" modules to more correctly reflect RFC 4279.

B.24. 22 to 23

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

B.25. 23 to 24

* Added missing reference to "FIPS PUB 180-4".

* Added identity "tls-1.3" and updated description statement in other identities indicating that the protocol version is obsolete and enabling the feature is NOT RECOMMENDED.

* Added XML-comment above examples explaining the reason for the unexpected top-most element's presence.

* Added missing "client-ident-raw-public-key" and "client-ident-psk" featutes.

* Aligned modules with `pyang -f` formatting.

* Fixed nits found by YANG Doctor reviews.

* Added a 'Contributors' section.

B.26. 24 to 25

* Added TLS 1.3 references.

* Clarified support for various TLS protocol versions.

* Moved algorithms in ietf-tls-common (plus more) to IANA-maintained modules

* Added "config false" lists for algorithms supported by the server.

* Fixed issues found during YANG Doctor review.

B.27. 25 to 26

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

B.28. 26 to 27

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)

* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

* Created identityref-based typedef for the IANA alg identity base.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balazs Kovacs, Benoit Claise, Bert Wijnen, David Lamparter, Dhruv Dhody, Gary Wu, Henk Birkholz, Juergen Schoenwaelder, Ladislav Lhotka, Liang Xia, Martin Bjoerklund, Mehmet Ersue, Michal Vasko, Phil Shafer, Radek Krejci, Sean Turner, and Tom Petch.

Contributors

Special acknowledgement goes to Gary Wu who contributed the "ietf-tls-common" module, and Tom Petch who carefully ensured that references were set correctly throughout.

Author’s Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

K. Watsen
Watsen Networks

7 March 2022

A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-17

Abstract

This document defines a YANG module for configuring bags of certificates and bags of public keys that can be referenced by other data models for trust. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types

* BBBB --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* 2022-03-07 --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 8 September 2022.

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 3
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
  1.4. Conventions 5
2. The "ietf-truststore" Module 6
  2.1. Data Model Overview 6
  2.2. Example Usage 12
  2.3. YANG Module 21
3. Support for Built-in Trust Anchors 29
4. Security Considerations 32
  4.1. Security of Data at Rest 32
  4.2. Unconstrained Public Key Usage 32
  4.3. The "ietf-truststore" YANG Module 32
5. IANA Considerations 33
  5.1. The "IETF XML" Registry 33
  5.2. The "YANG Module Names" Registry 33
6. References 33
  6.1. Normative References 33
  6.2. Informative References 34

Appendix A. Change Log 36
A.1. 00 to 01 36
A.2. 01 to 02 36
A.3. 02 to 03 36
A.4. 03 to 04 36
A.5. 04 to 05 37
A.6. 05 to 06 37
A.7. 06 to 07 37
A.8. 07 to 08 37
A.9. 08 to 09 37
A.10. 09 to 10 38
A.11. 10 to 11 38
A.12. 11 to 12 38
A.13. 12 to 13 38
A.14. 13 to 14 38
A.15. 14 to 15 38
A.16. 15 to 16 39
A.17. 16 to 17 39
Acknowledgements 39
Author's Address 39

1. Introduction

This document defines a YANG 1.1 [RFC7950] module having the following characteristics:

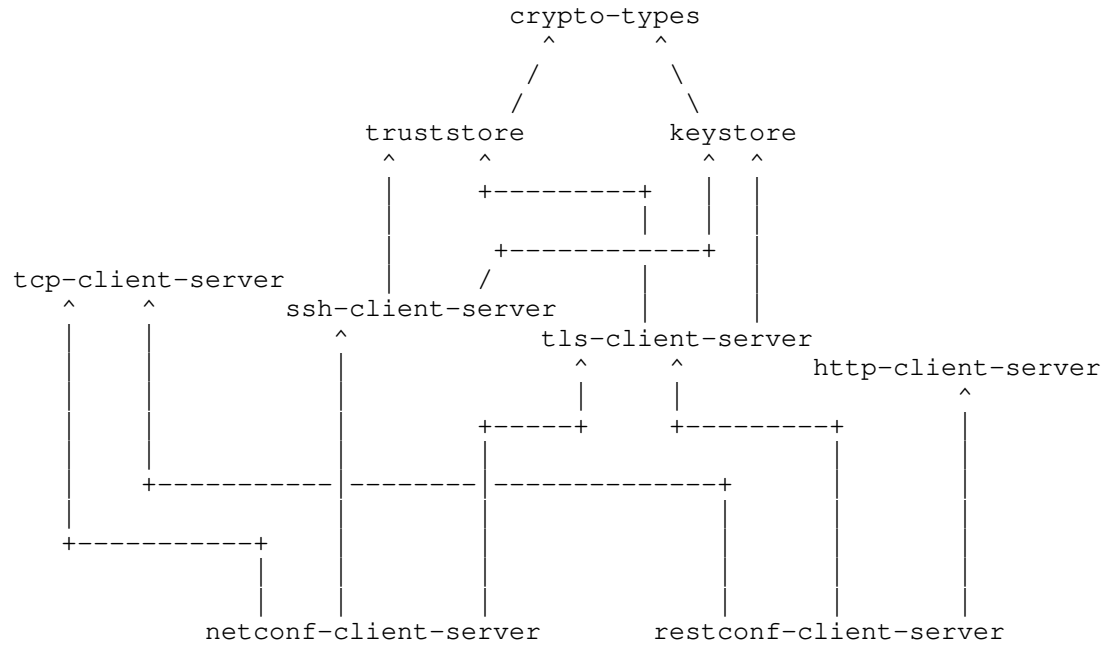
- Provide a central truststore for storing raw public keys and/or certificates.
- Provide support for storing named bags of raw public keys and/or named bags of certificates.
- Provide types that can be used to reference raw public keys or certificates stored in the central truststore.
- Provide groupings that enable raw public keys and certificates to be configured locally or as references truststore instances.
- Enable the truststore to be instantiated in other data models, in addition to or in lieu of the central trusttore instance.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, trust anchors installed during manufacturing (e.g., for trusted well-known services), are expected to appear in <operational> (see Section 3).

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-truststore" Module

This section defines a YANG 1.1 [RFC7950] module that defines a "truststore" and groupings supporting downstream modules to reference the truststore or have locally-defined definitions.

This section defines a YANG 1.1 [RFC7950] module called "ietf-truststore". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-truststore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-truststore" module:

```
Features:
  +-- central-truststore-supported
  +-- local-definitions-supported
  +-- certificates
  +-- public-keys
```

```
  |  The diagram above uses syntax that is similar to but not
  |  defined in [RFC8340].
```

2.1.2. Typedefs

The following diagram lists the "typedef" statements defined in the "ietf-truststore" module:

```
Typedefs:
  leafref
    +-- certificate-bag-ref
    +-- certificate-ref
    +-- public-key-bag-ref
    +-- public-key-ref
```

```
  |  The diagram above uses syntax that is similar to but not
  |  defined in [RFC8340].
```

Comments:

* All the typedefs defined in the "ietf-truststore" module extend the base "leafref" type defined in [RFC7950].

* The leafrefs refer to certificates, public keys, and bags in the central truststore, when this module is implemented.

* These typedefs are provided as an aid to downstream modules that import the "ietf-truststore" module.

2.1.3. Groupings

The "ietf-truststore" module defines the following "grouping" statements:

* local-or-truststore-certs-grouping
* local-or-truststore-public-keys-grouping
* truststore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "local-or-truststore-certs-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-truststore-certs-grouping" grouping:

```
  grouping local-or-truststore-certs-grouping
    +-- (local-or-truststore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +-- certificate* [name]
       |        +-- name?                              string
       |        +---u ct:trust-anchor-cert-grouping
       +--:(truststore) {central-truststore-supported,certificates}?
          +-- truststore-reference?   ts:certificate-bag-ref
```

Comments:

* The "local-or-truststore-certs-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option whether a bag of certificates can be defined locally or as a reference to a bag in the truststore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a bag in an alternate location.

* For the "local-definition" option, the "certificate" node uses the "trust-anchor-cert-grouping" grouping discussed in Section 2.1.4.7 of [I-D.ietf-netconf-crypto-types].

* For the "truststore" option, the "truststore-reference" is an instance of the "certificate-bag-ref" discussed in Section 2.1.2.

2.1.3.2. The "local-or-truststore-public-keys-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-truststore-public-keys-grouping" grouping:

```
  grouping local-or-truststore-public-keys-grouping
    +-- (local-or-truststore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +-- public-key* [name]
       |        +-- name?                      string
       |        +---u ct:public-key-grouping
       +--:(truststore) {central-truststore-supported,public-keys}?
          +-- truststore-reference?   ts:public-key-bag-ref
```

Comments:

* The "local-or-truststore-public-keys-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option whether a bag of public keys can be defined locally or as a reference to a bag in the truststore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a bag in an alternate location.

* For the "local-definition" option, the "public-key" node uses the "public-key-grouping" grouping discussed in Section 2.1.4.4 of [I-D.ietf-netconf-crypto-types].

* For the "truststore" option, the "truststore-reference" is an instance of the "certificate-bag-ref" discussed in Section 2.1.2.

2.1.3.3. The "truststore-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "truststore-grouping" grouping:

```
  grouping truststore-grouping
    +-- certificate-bags {certificates}?
    |  +-- certificate-bag* [name]
    |     +-- name?          string
    |     +-- description?   string
    |     +-- certificate* [name]
    |        +-- name?                              string
    |        +---u ct:trust-anchor-cert-grouping
    +-- public-key-bags {public-keys}?
       +-- public-key-bag* [name]
          +-- name?          string
          +-- description?   string
          +-- public-key* [name]
             +-- name?                      string
             +---u ct:public-key-grouping
```

Comments:

* The "truststore-grouping" grouping defines a truststore instance as being composed of certificates and/or public keys, both of which are enabled by "feature" statements. The structure supporting certificates and public keys is essentially the same, having an outer list of "bags" containing in inner list of objects (certificates or public keys). The bags enable trust anchors serving a common purpose to be grouped and referenced together.

* For certificates, each certificate is defined by the "trust-anchor-cert-grouping" grouping Section 2.1.4.7 of [I-D.ietf-netconf-crypto-types]. Thus the "cert-data" node is a CMS structure that can be composed of a chain of one or more certificates. Additionally, the "certificate-expiration" notification enables the server to alert clients when certificates are nearing or have already expired.

* For public keys, each public key is defined by the "public-key-grouping" grouping Section 2.1.4.4 of [I-D.ietf-netconf-crypto-types]. Thus the "public-key" node can be one of any number of structures specified by the "public-key-format" identity node.

2.1.4. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-truststore" module, without expanding the "grouping" statements:

```
module: ietf-truststore
  +--rw truststore
     +---u truststore-grouping

  grouping local-or-truststore-certs-grouping
    +-- (local-or-truststore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +-- certificate* [name]
       |        +-- name?                              string
       |        +---u ct:trust-anchor-cert-grouping
       +--:(truststore) {central-truststore-supported,certificates}?
          +-- truststore-reference?   ts:certificate-bag-ref
  grouping local-or-truststore-public-keys-grouping
    +-- (local-or-truststore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +-- public-key* [name]
       |        +-- name?                       string
       |        +---u ct:public-key-grouping
       +--:(truststore) {central-truststore-supported,public-keys}?
          +-- truststore-reference?   ts:public-key-bag-ref
  grouping truststore-grouping
    +-- certificate-bags {certificates}?
    |  +-- certificate-bag* [name]
    |     +-- name?          string
    |     +-- description?   string
    |     +-- certificate* [name]
    |        +-- name?                              string
    |        +---u ct:trust-anchor-cert-grouping
    +-- public-key-bags {public-keys}?
       +-- public-key-bag* [name]
          +-- name?          string
          +-- description?   string
          +-- public-key* [name]
             +-- name?                       string
             +---u ct:public-key-grouping
```

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-truststore" module, with all "grouping" statements expanded, enabling the truststore's full structure to be seen:

```
module: ietf-truststore
  +--rw truststore
     +--rw certificate-bags {certificates}?
     |  +--rw certificate-bag* [name]
     |     +--rw name           string
     |     +--rw description?   string
     |     +--rw certificate* [name]
     |        +--rw name                      string
     |        +--rw cert-data                 trust-anchor-cert-cms
     |        +---n certificate-expiration
     |                {certificate-expiration-notification}?
     |           +-- expiration-date    yang:date-and-time
     +--rw public-key-bags {public-keys}?
        +--rw public-key-bag* [name]
           +--rw name           string
           +--rw description?   string
           +--rw public-key* [name]
              +--rw name                 string
              +--rw public-key-format    identityref
              +--rw public-key           binary

  grouping local-or-truststore-certs-grouping
    +-- (local-or-truststore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +-- certificate* [name]
       |        +-- name?                       string
       |        +-- cert-data                   trust-anchor-cert-cms
       |        +---n certificate-expiration
       |                {certificate-expiration-notification}?
       |           +-- expiration-date    yang:date-and-time
       +--:(truststore) {central-truststore-supported,certificates}?
          +-- truststore-reference?   ts:certificate-bag-ref
  grouping local-or-truststore-public-keys-grouping
    +-- (local-or-truststore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +-- public-key* [name]
       |        +-- name?                 string
       |        +-- public-key-format     identityref
       |        +-- public-key            binary
       +--:(truststore) {central-truststore-supported,public-keys}?
          +-- truststore-reference?   ts:public-key-bag-ref
  grouping truststore-grouping
    +-- certificate-bags {certificates}?
    |  +-- certificate-bag* [name]
    |     +-- name?          string
    |     +-- description?   string
```

```
      |     +-- certificate* [name]
      |        +-- name?                    string
      |        +-- cert-data                trust-anchor-cert-cms
      |        +---n certificate-expiration
      |                {certificate-expiration-notification}?
      |           +-- expiration-date    yang:date-and-time
      +-- public-key-bags {public-keys}?
         +-- public-key-bag* [name]
            +-- name?          string
            +-- description?   string
            +-- public-key* [name]
               +-- name?                string
               +-- public-key-format    identityref
               +-- public-key           binary
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

* The protcol-accessible nodes for the "ietf-truststore" module are an instance of the "truststore-grouping" grouping discussed in Section 2.1.3.3.

* The reason for why the "truststore-grouping" exists separate from the protocol-accessible nodes definition is to enable instances of the truststore to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The examples in this section are encoded using XML, such as might be the case when using the NETCONF protocol. Other encodings MAY be used, such as JSON when using the RESTCONF protocol.

2.2.1. A Truststore Instance

This section presents an example illustrating trust anchors in <intended>, as per Section 2.1.4. Please see Section 3 for an example illustrating built-in values in <operational>.

The example contained in this section defines eight bags of trust anchors. There are four certificate-based bags and four public key based bags. The following diagram provides an overview of the contents in the example:

```
   Certificate Bags
     +-- Trust anchor certs for authenticating a set of remote servers
     +-- End entity certs for authenticating a set of remote servers
     +-- Trust anchor certs for authenticating a set of remote clients
     +-- End entity certs for authenticating a set of remote clients

   Public Key Bags
     +-- SSH keys to authenticate a set of remote SSH server
     +-- SSH keys to authenticate a set of remote SSH clients
     +-- Raw public keys to authenticate a set of remote SSH server
     +-- Raw public keys to authenticate a set of remote SSH clients
```

Following is the full example:

```
   <truststore
     xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
     xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

     <!-- A bag of Certificate Bags -->
     <certificate-bags>

       <!-- Trust Anchor Certs for Authenticating Servers -->
       <certificate-bag>
         <name>trusted-server-ca-certs</name>
         <description>
           Trust anchors (i.e. CA certs) used to authenticate server
           certificates.  A server certificate is authenticated if its
           end-entity certificate has a chain of trust to one of these
           certificates.
         </description>
         <certificate>
           <name>Server Cert Issuer #1</name>
           <cert-data>BASE64VALUE=</cert-data>
         </certificate>
         <certificate>
           <name>Server Cert Issuer #2</name>
           <cert-data>BASE64VALUE=</cert-data>
         </certificate>
       </certificate-bag>

       <!-- End Entity Certs for Authenticating Servers -->
       <certificate-bag>
         <name>trusted-server-ee-certs</name>
         <description>
           Specific end-entity certificates used to authenticate server
           certificates.  A server certificate is authenticated if its
           end-entity certificate is an exact match to one of these
           certificates.
```

```
          </description>
          <certificate>
            <name>My Application #1</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>My Application #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </certificate-bag>

        <!-- Trust Anchor Certs for Authenticating Clients -->
        <certificate-bag>
          <name>trusted-client-ca-certs</name>
          <description>
            Trust anchors (i.e. CA certs) used to authenticate client
            certificates.  A client certificate is authenticated if its
            end-entity certificate has a chain of trust to one of these
            certificates.
          </description>
          <certificate>
            <name>Client Identity Issuer #1</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>Client Identity Issuer #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </certificate-bag>

        <!-- End Entity Certs for Authenticating Clients -->
        <certificate-bag>
          <name>trusted-client-ee-certs</name>
          <description>
            Specific end-entity certificates used to authenticate client
            certificates.  A client certificate is authenticated if its
            end-entity certificate is an exact match to one of these
            certificates.
          </description>
          <certificate>
            <name>George Jetson</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>Fred Flintstone</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </certificate-bag>
```

```
    </certificate-bags>

    <!-- A List of Public Key Bags -->
    <public-key-bags>

      <!-- Public Keys for Authenticating SSH Servers -->
      <public-key-bag>
        <name>trusted-ssh-public-keys</name>
        <description>
          Specific SSH public keys used to authenticate SSH server
          public keys.  An SSH server public key is authenticated if
          its public key is an exact match to one of these public keys.

          This list of SSH public keys is analogous to OpenSSH's
          "/etc/ssh/ssh_known_hosts" file.
        </description>
        <public-key>
          <name>corp-fw1</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>corp-fw2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>

      <!-- SSH Public Keys for Authenticating Application A -->
      <public-key-bag>
        <name>SSH Public Keys for Application A</name>
        <description>
          SSH public keys used to authenticate application A's SSH
          public keys.  An SSH public key is authenticated if it
          is an exact match to one of these public keys.
        </description>
        <public-key>
          <name>Application Instance #1</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
```

```
          <name>Application Instance #2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>

      <!-- Raw Public Keys for TLS Servers -->
      <public-key-bag>
        <name>Raw Public Keys for TLS Servers</name>
        <public-key>
          <name>Raw Public Key #1</name>
          <public-key-format>
            ct:subject-public-key-info-format</public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>Raw Public Key #2</name>
          <public-key-format>
            ct:subject-public-key-info-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>

      <!-- Raw Public Keys for TLS Clients -->
      <public-key-bag>
        <name>Raw Public Keys for TLS Clients</name>
        <public-key>
          <name>Raw Public Key #1</name>
          <public-key-format>
            ct:subject-public-key-info-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>Raw Public Key #2</name>
          <public-key-format>
            ct:subject-public-key-info-format
        </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>
    </public-key-bags>
  </truststore>
```

2.2.2. A Certificate Expiration Notification

The following example illustrates the "certificate-expiration" notification (per Section 2.1.4.6 of [I-D.ietf-netconf-crypto-types]) for a certificate configured in the truststore in Section 2.2.1.

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
    <certificate-bags>
      <certificate-bag>
        <name>trusted-client-ee-certs</name>
        <certificate>
          <name>George Jetson</name>
          <certificate-expiration>
            <expiration-date>2018-08-05T14:18:53-05:00</expiration-d\
ate>
          </certificate-expiration>
        </certificate>
      </certificate-bag>
    </certificate-bags>
  </truststore>
</notification>
```

2.2.3. The "Local or Truststore" Groupings

This section illustrates the various "local-or-truststore" groupings defined in the "ietf-truststore" module, specifically the "local-or-truststore-certs-grouping" (Section 2.1.3.1) and "local-or-truststore-public-keys-grouping" (Section 2.1.3.2) groupings.

These examples assume the existence of an example module called "ex-truststore-usage" having the namespace "http://example.com/ns/example-truststore-usage".

The ex-truststore-usage module is first presented using tree diagrams [RFC8340], followed by an instance example illustrating all the "local-or-truststore" groupings in use, followed by the YANG module itself.

The following tree diagram illustrates "ex-truststore-usage" without expanding the "grouping" statements:

```
module: ex-truststore-usage
  +--rw truststore-usage
     +--rw cert* [name]
     |  +--rw name                                        string
     |  +---u ts:local-or-truststore-certs-grouping
     +--rw public-key* [name]
        +--rw name                                            string
        +---u ts:local-or-truststore-public-keys-grouping
```

The following tree diagram illustrates the "ex-truststore-usage" module, with all "grouping" statements expanded, enabling the truststore's full structure to be seen:

```
module: ex-truststore-usage
  +--rw truststore-usage
     +--rw cert* [name]
     |  +--rw name                           string
     |  +--rw (local-or-truststore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw certificate* [name]
     |     |        +--rw name                      string
     |     |        +--rw cert-data
     |     |        |       trust-anchor-cert-cms
     |     |        +---n certificate-expiration
     |     |                 {certificate-expiration-notification}?
     |     |           +-- expiration-date    yang:date-and-time
     |     +--:(truststore)
     |              {central-truststore-supported,certificates}?
     |        +--rw truststore-reference?   ts:certificate-bag-ref
     +--rw public-key* [name]
        +--rw name                           string
        +--rw (local-or-truststore)
           +--:(local) {local-definitions-supported}?
           |  +--rw local-definition
           |     +--rw public-key* [name]
           |        +--rw name                 string
           |        +--rw public-key-format    identityref
           |        +--rw public-key           binary
           +--:(truststore)
                    {central-truststore-supported,public-keys}?
              +--rw truststore-reference?   ts:public-key-bag-ref
```

The following example provides two equivalent instances of each grouping, the first being a reference to a truststore and the second being locally-defined. The instance having a reference to a truststore is consistent with the truststore defined in Section 2.2.1. The two instances are equivalent, as the locally-defined instance example contains the same values defined by the truststore instance referenced by its sibling example.

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<truststore-usage
  xmlns="http://example.com/ns/example-truststore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- The following two equivalent examples illustrate   -->
  <!-- the "local-or-truststore-certs-grouping" grouping: -->

  <cert>
    <name>example 1a</name>
    <truststore-reference>trusted-client-ca-certs</truststore-refere\
nce>
  </cert>

  <cert>
    <name>example 1b</name>
    <local-definition>
      <name>my-trusted-client-ca-certs</name>
      <certificate>
        <name>Client Identity Issuer #1</name>
        <cert>BASE64VALUE=</cert>
      </certificate>
      <certificate>
        <name>Client Identity Issuer #2</name>
        <cert>BASE64VALUE=</cert>
      </certificate>
    </local-definition>
  </cert>


  <!-- The following two equivalent examples illustrate the -->
  <!-- "local-or-truststore-public-keys-grouping" grouping: -->

  <public-key>
    <name>example 2a</name>
    <truststore-reference>trusted-ssh-public-keys</truststore-refere\
nce>
  </public-key>
```

```
    <public-key>
      <name>example 2b</name>
      <local-definition>
        <name>trusted-ssh-public-keys</name>
        <public-key>
          <name>corp-fw1</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>corp-fw2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </local-definition>
    </public-key>

  </truststore-usage>
```

Following is the "ex-truststore-usage" module's YANG definition:

```
  module ex-truststore-usage {
    yang-version 1.1;
    namespace "http://example.com/ns/example-truststore-usage";
    prefix etu;

    import ietf-truststore {
      prefix ts;
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

    organization
      "Example Corporation";

    contact
      "Author: YANG Designer <mailto:yang.designer@example.com>";

    description
      "This module illustrates notable groupings defined in
       the 'ietf-truststore' module.";

    revision 2022-03-07 {
      description
```

```
        "Initial version";
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

    container truststore-usage {
      description
        "An illustration of the various truststore groupings.";
      list cert {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this cert.";
        }
        uses ts:local-or-truststore-certs-grouping;
        description
          "An cert that may be configured locally or be
           a reference to a cert in the truststore.";
      }
      list public-key {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this cert.";
        }
        uses ts:local-or-truststore-public-keys-grouping;
        description
          "An public key that may be configured locally or be
           a reference to a public key in the truststore.";
      }
    }
  }
```

2.3. YANG Module

This YANG module imports modules from [RFC8341] and [I-D.ietf-netconf-crypto-types].

<CODE BEGINS> file "ietf-truststore@2022-03-07.yang"

```
module ietf-truststore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-truststore";
  prefix ts;

  import ietf-netconf-acm {
```

```
      prefix nacm;
      reference
        "RFC 8341: Network Configuration Access Control Model";
    }

    import ietf-crypto-types {
      prefix ct;
      reference
        "RFC AAAA: YANG Data Types and Groupings for Cryptography";
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group";

    contact
      "WG Web  : https://datatracker.ietf.org/wg/netconf
       WG List : NETCONF WG list <mailto:netconf@ietf.org>
       Author  : Kent Watsen <kent+ietf@watsen.net>";

    description
      "This module defines a 'truststore' to centralize management
       of trust anchors including certificates and public keys.

       Copyright (c) 2021 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

       Redistribution and use in source and binary forms, with
       or without modification, is permitted pursuant to, and
       subject to the license terms contained in, the Revised
       BSD License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC BBBB
       (https://www.rfc-editor.org/info/rfcBBBB); see the RFC
       itself for full legal notices.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
       'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
       'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
       are to be interpreted as described in BCP 14 (RFC 2119)
       (RFC 8174) when, and only when, they appear in all
       capitals, as shown here.";

    revision 2022-03-07 {
      description
        "Initial version";
      reference
```

```
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

    /****************/
    /*   Features   */
    /****************/

    feature central-truststore-supported {
      description
        "The 'central-truststore-supported' feature indicates that
         the server supports the truststore (i.e., implements the
         'ietf-truststore' module).";
    }

    feature local-definitions-supported {
      description
        "The 'local-definitions-supported' feature indicates that
         the server supports locally-defined trust anchors.";
    }

    feature certificates {
      description
        "The 'certificates' feature indicates that the server
         implements the /truststore/certificate-bags subtree.";
    }

    feature public-keys {
      description
        "The 'public-keys' feature indicates that the server
         implements the /truststore/public-key-bags subtree.";
    }

    /****************/
    /*   Typedefs   */
    /****************/

    typedef certificate-bag-ref {
      type leafref {
        path "/ts:truststore/ts:certificate-bags/"
           + "ts:certificate-bag/ts:name";
      }
      description
        "This typedef defines a reference to a certificate bag
         in the truststore, when this module is implemented.";
    }

    typedef certificate-ref {
      type leafref {
```

```
        path "/ts:truststore/ts:certificate-bags/ts:certificate-bag"
           + "[ts:name = current()/../ts:certificate-bag]/"
           + "ts:certificate/ts:name";
      }
      description
        "This typedef defines a reference to a specific certificate
         in a certificate bag in the truststore, when this module
         is implemented.  This typedef requires that there exist a
         sibling 'leaf' node called 'certificate-bag' that SHOULD
         have the typedef 'certificate-bag-ref'.";
    }

    typedef public-key-bag-ref {
      type leafref {
        path "/ts:truststore/ts:public-key-bags/"
           + "ts:public-key-bag/ts:name";
      }
      description
        "This typedef defines a reference to a public key bag
         in the truststore, when this module is implemented.";
    }

    typedef public-key-ref {
      type leafref {
        path "/ts:truststore/ts:public-key-bags/ts:public-key-bag"
           + "[ts:name = current()/../ts:public-key-bag]/"
           + "ts:public-key/ts:name";
      }
      description
        "This typedef defines a reference to a specific public key
         in a public key bag in the truststore, when this module is
         implemented.  This typedef requires that there exist a
         sibling 'leaf' node called 'public-key-bag' that SHOULD
         have the typedef 'public-key-bag-ref'.";
    }

    /*****************/
    /*   Groupings   */
    /*****************/

    grouping local-or-truststore-certs-grouping {
      description
        "A grouping that allows the certificates to be either
         configured locally, within the using data model, or be a
         reference to a certificate bag stored in the truststore.

         Servers that do not 'implement' this module, and hence
         'central-truststore-supported' is not defined, SHOULD
```

```
          augment in custom 'case' statements enabling references
          to the alternate truststore locations.";
        choice local-or-truststore {
          nacm:default-deny-write;
          mandatory true;
          description
            "A choice between an inlined definition and a definition
             that exists in the truststore.";
          case local {
            if-feature "local-definitions-supported";
            container local-definition {
              description
                "A container for locally configured trust anchor
                 certificates.";
              list certificate {
                key "name";
                min-elements 1;
                description
                  "A trust anchor certificate.";
                leaf name {
                  type string;
                  description
                    "An arbitrary name for this certificate.";
                }
                uses ct:trust-anchor-cert-grouping {
                  refine "cert-data" {
                    mandatory true;
                  }
                }
              }
            }
          }
          case truststore {
            if-feature "central-truststore-supported";
            if-feature "certificates";
            leaf truststore-reference {
              type ts:certificate-bag-ref;
              description
                "A reference to a certificate bag that exists in the
                 truststore, when this module is implemented.";
            }
          }
        }
      }

      grouping local-or-truststore-public-keys-grouping {
        description
          "A grouping that allows the public keys to be either
```

```
          configured locally, within the using data model, or be a
          reference to a public key bag stored in the truststore.

          Servers that do not 'implement' this module, and hence
          'central-truststore-supported' is not defined, SHOULD
          augment in custom 'case' statements enabling references
          to the alternate truststore locations.";
        choice local-or-truststore {
          nacm:default-deny-write;
          mandatory true;
          description
            "A choice between an inlined definition and a definition
             that exists in the truststore.";
          case local {
            if-feature "local-definitions-supported";
            container local-definition {
              description
                "A container to hold local public key definitions.";
              list public-key {
                key "name";
                description
                  "A public key definition.";
                leaf name {
                  type string;
                  description
                    "An arbitrary name for this public key.";
                }
                uses ct:public-key-grouping;
              }
            }
          }
          case truststore {
            if-feature "central-truststore-supported";
            if-feature "public-keys";
            leaf truststore-reference {
              type ts:public-key-bag-ref;
              description
                "A reference to a bag of public keys that exists
                 in the truststore, when this module is implemented.";
            }
          }
        }
      }

      grouping truststore-grouping {
        description
          "A grouping definition that enables use in other contexts.
           Where used, implementations MUST augment new 'case'
```

```
          statements into the various local-or-truststore 'choice'
          statements to supply leafrefs to the model-specific
          location(s).";
        container certificate-bags {
          nacm:default-deny-write;
          if-feature "certificates";
          description
            "A collection of certificate bags.";
          list certificate-bag {
            key "name";
            description
              "A bag of certificates.  Each bag of certificates SHOULD
               be for a specific purpose.  For instance, one bag could
               be used to authenticate a specific set of servers, while
               another could be used to authenticate a specific set of
               clients.";
            leaf name {
              type string;
              description
                "An arbitrary name for this bag of certificates.";
            }
            leaf description {
              type string;
              description
                "A description for this bag of certificates.  The
                 intended purpose for the bag SHOULD be described.";
            }
            list certificate {
              key "name";
              description
                "A trust anchor certificate.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this certificate.";
              }
              uses ct:trust-anchor-cert-grouping {
                refine "cert-data" {
                  mandatory true;
                }
              }
            }
          }
        }
        container public-key-bags {
          nacm:default-deny-write;
          if-feature "public-keys";
          description
```

```
          "A collection of public key bags.";
        list public-key-bag {
          key "name";
          description
            "A bag of public keys.  Each bag of keys SHOULD be for
             a specific purpose.  For instance, one bag could be used
             authenticate a specific set of servers, while another
             could be used to authenticate a specific set of clients.";
          leaf name {
            type string;
            description
              "An arbitrary name for this bag of public keys.";
          }
          leaf description {
            type string;
            description
              "A description for this bag public keys.  The
               intended purpose for the bag SHOULD be described.";
          }
          list public-key {
            key "name";
            description
              "A public key.";
            leaf name {
              type string;
              description
                "An arbitrary name for this public key.";
            }
            uses ct:public-key-grouping;
          }
        }
      }
    }

    /*********************************/
    /*   Protocol accessible nodes   */
    /*********************************/

    container truststore {
      nacm:default-deny-write;
      description
        "The truststore contains bags of certificates and
         public keys.";
      uses truststore-grouping;
    }
  }

  <CODE ENDS>
```

3. Support for Built-in Trust Anchors

In some implementations, a server may define some built-in trust anchors. For instance, there may be built-in trust anchors enabling the server to securely connect to well-known services (e.g., an SZTP [RFC8572] bootstrap server) or public CA certificates to connect to arbitrary services using public PKI.

Built-in trust anchors are expected to be set by a vendor-specific process. Any ability for operators to modify built-in trust anchors is outside the scope of this document.

As built-in trust anchors are provided by the server, they are present in <operational> (and <system> [I-D.ma-netmod-with-system], if used). The example below illustrates what the truststore in <operational> might look like for a server in its factory default state.

```
<truststore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <certificate-bags>

    <certificate-bag or:origin="or:system">
      <name>Built-In Manufacturer Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        manufacturer-signed objects, such as TLS server certificates,
        vouchers, etc.
      </description>
      <certificate>
        <name>Manufacturer Root CA Cert</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

    <certificate-bag or:origin="or:system">
      <name>Built-In Public Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        certificates issued by public certificate authorities,
        such as the end-entity certificate for web servers.
      </description>
      <certificate>
        <name>Public Root CA Cert 1</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
      <certificate>
        <name>Public Root CA Cert 2</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
      <certificate>
        <name>Public Root CA Cert 3</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

  </certificate-bags>
</truststore>
```

In order for the built-in bags of trust anchors and/or their trust anchors to be referenced by configuration, they MUST first be copied into <running>.

The built-in bags and/or their trust anchors MUST be copied into <running> using the same "key" values if it is desired for the server to maintain/update them (e.g., a software update may update a bag of trusted public CA certificates used for TLS-client connections).

Built-in bags and/or their trust anchors MAY be copied into other parts of the configuration but, by doing so, they lose their association to the built-in entries and any assurances afforded by knowing they are/were built-in.

The built-in bags and/or their trust anchors are immutable by configuration operations. Servers MUST ignore attempts to modify any aspect of built-in bags and/or their trust anchors from <running>.

The following example illustrates how a single built-in public CA certificate from the previous example has been propagated to <running>:

```
<truststore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <certificate-bags>

    <certificate-bag>
      <name>Built-In Public Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        certificates issued by public certificate authorities,
        such as the end-entity certificate for web servers.

        Only the subset of the certificates that are referenced
        by other configuration nodes need to be copied.  For
        instance, only "Public Root CA Cert 3" is present here.

        No new certificates can be added, nor existing certificate
        values changed.  Missing certificates have no effect on
        "operational" when the configuration is applied.
      </description>
      <certificate>
        <name>Public Root CA Cert 3</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

  </certificate-bags>
</truststore>
```

4. Security Considerations

4.1. Security of Data at Rest

The YANG module defined in this document defines a mechanism called a "truststore" that, by its name, suggests that its contents are protected from unauthorized modification.

Security controls for the API (i.e., data in motion) are discussed in Section 4.3, but controls for the data at rest cannot be specified by the YANG module.

In order to satisfy the expectations of a "truststore", it is RECOMMENDED that implementations ensure that the truststore contents are protected from unauthorized modifications when at rest.

4.2. Unconstrained Public Key Usage

This module enables the configuration of public keys without constraints on their usage, e.g., what operations the key is allowed to be used for (encryption, verification, both).

This module also enables the configuration of certificates, where each certificate may constrain the usage of the public key according to local policy.

4.3. The "ietf-truststore" YANG Module

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

All the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "truststore" instance, may be considered sensitive or vulnerable in some network environments. For instance, any modification to a trust anchor or reference to a trust anchor may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any "rpc" or "action" statements, and thus the security considerations for such is not provided here.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-truststore
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

5.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registration is requested:

```
name:         ietf-truststore
namespace:    urn:ietf:params:xml:ns:yang:ietf-truststore
prefix:       ts
reference:    RFC BBBB
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

6.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>.

[I-D.ma-netmod-with-system]
Ma, Q., Watsen, K., Wu, Q., Chong, F., and J. Lindblad, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ma-netmod-with-system-02, 14 February 2022, <https://datatracker.ietf.org/doc/html/draft-ma-netmod-with-system-02>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Added features "x509-certificates" and "ssh-host-keys".

* Added nacm:default-deny-write to "trust-anchors" container.

A.2. 01 to 02

* Switched "list pinned-certificate" to use the "trust-anchor-cert-grouping" from crypto-types. Effectively the same definition as before.

A.3. 02 to 03

* Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

A.4. 03 to 04

* Added groupings 'local-or-truststore-certs-grouping' and 'local-or-truststore-host-keys-grouping', matching similar definitions in the keystore draft. Note new (and incomplete) "truststore" usage!

* Related to above, also added features 'truststore-supported' and 'local-trust-anchors-supported'.

A.5. 04 to 05

* Renamed "trust-anchors" to "truststore"
* Removed "pinned." prefix everywhere, to match truststore rename
* Moved everything under a top-level 'grouping' to enable use in other contexts.
* Renamed feature from 'local-trust-anchors-supported' to 'local-definitions-supported' (same name used in keystore)
* Removed the "require-instance false" statement from the "*-ref" typedefs.
* Added missing "ssh-host-keys" and "x509-certificates" if-feature statements

A.6. 05 to 06

* Editorial changes only.

A.7. 06 to 07

* Added Henk Birkholz as a co-author (thanks Henk!)
* Added PSKs and raw public keys to truststore.

A.8. 07 to 08

* Added new "Support for Built-in Trust Anchors" section.
* Removed spurious "uses ct:trust-anchor-certs-grouping" line.
* Removed PSK from model.

A.9. 08 to 09

* Removed remaining PSK references from text.
* Wrapped each top-level list with a container.
* Introduced "bag" term.
* Merged "SSH Public Keys" and "Raw Public Keys" in a single "Public Keys" bag. Consuming downstream modules (i.e., "ietf-[ssh/tls]-[client/server]) refine the "public-key-format" to be either SSH or TLS specific as needed.

A.10. 09 to 10

* Removed "algorithm" node from examples.
* Removed the no longer used statements supporting the old "ssh-public-key" and "raw-public-key" nodes.
* Added a "Note to Reviewers" note to first page.

A.11. 10 to 11

* Corrected module prefix registered in the IANA Considerations section.
* Modified 'local-or-truststore-certs-grouping' to use a list (not a leaf-list).
* Added new example section "The Local or Truststore Groupings".
* Clarified expected behavior for "built-in" certificates in <operational>
* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
* Updated the Security Considerations section.

A.12. 11 to 12

* Fixed a copy/paste issue in the "Data at Rest" Security Considerations section.

A.13. 12 to 13

* Fixed issues found by the SecDir review of the "keystore" draft.

A.14. 13 to 14

* Added an "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir.
* Addressed comments raised by YANG Doctor.

A.15. 14 to 15

* Added prefixes to 'path' statements per trust-anchors/issues/1

* Renamed feature "truststore-supported" to "central-truststore-supported".

* Associated with above, generally moved text to refer to a "central" truststore.

* Removed two unecessary/unwanted "min-elements 1" and associated "presence" statements.

* Aligned modules with ‘pyang -f‘ formatting.

* Fixed nits found by YANG Doctor reviews.

A.16. 15 to 16

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

A.17. 16 to 17

* fixup the ’WG Web’ and ’WG List’ lines in YANG module(s)

* fixup copyright (i.e., s/Simplified/Revised/) in YANG module(s)

* Added Informative reference to ma-netmod-with-system

Acknowledgements

The authors especially thank Henk Birkholz for contributing YANG to the ietf-truststore module supporting raw public keys and PSKs (pre-shared or pairwise-symmetric keys). While these contributions were eventually replaced by reusing the existing support for asymmetric and symmetric trust anchors, respectively, it was only thru Henk’s initiative that the WG was able to come to that result.

The authors additionally thank the following for helping give shape to this work (ordered by first name): Balazs Kovacs, Eric Voit, Juergen Schoenwaelder, Liang Xia, Martin Bjoerklund, Nick Hancock, and Yoav Nir.

Author’s Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: 4 September 2022

G. Zheng
T. Zhou
Huawei
T. Graf
Swisscom
P. Francois
A. Huang Feng
INSA-Lyon
P. Lucente
NTT

3 March 2022

UDP-based Transport for Configured Subscriptions
draft-ietf-netconf-udp-notif-05

Abstract

This document describes an UDP-based notification mechanism to collect data from networking devices. A shim header is proposed to facilitate the data streaming directly from the publishing process on network processor of line cards to receivers. The objective is to provide a lightweight approach to enable higher frequency and less performance impact on publisher and receiver processes compared to already established notification mechanisms.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 4 September 2022.

Table of Contents

1. Introduction 3
2. Configured Subscription to UDP-Notif 4
3. UDP-Based Transport 4
3.1. Design Overview 5
3.2. Format of the UDP-Notif Message Header 5
3.3. Data Encoding 7
4. Options 7
4.1. Segmentation Option 8
4.2. Private Encoding Option 9
5. Applicability 9
5.1. Congestion Control 10
5.2. Message Size 10
5.3. Reliability 11
5.4. Security Considerations 11
6. Secured layer for UDP-notif 11
6.1. Transport 12
6.2. Port Assignment 13
6.3. Session lifecycle 13
6.3.1. DTLS Session Initiation 13
6.3.2. Publish Data 13
6.3.3. Session termination 14
7. A YANG Data Model for Management of UDP-Notif 14
8. YANG Module 15
9. IANA Considerations 18
10. Acknowledgements 19
11. References 19
11.1. Normative References 19
11.2. Informative References 20
Authors' Addresses 21

1. Introduction

Sub-Notif [RFC8639] defines a mechanism that lets a receiver subscribe to the publication of YANG-defined data maintained in a YANG [RFC7950] datastore. The mechanism separates the management and control of subscriptions from the transport used to deliver the data. Three transport mechanisms, namely NETCONF transport [RFC8640], RESTCONF transport [RFC8650], and HTTPS transport [I-D.ietf-netconf-https-notif] have been defined so far for such notification messages.

While powerful in their features and general in their architecture, the currently available transport mechanisms need to be complemented to support data publications at high velocity from devices that feature a distributed architecture. The currently available transports are based on TCP and lack the efficiency needed to continuously send notifications at high velocity.

This document specifies a transport option for Sub-Notif that leverages UDP. Specifically, it facilitates the distributed data collection mechanism described in [I-D.ietf-netconf-distributed-notif]. In the case of publishing from multiple network processors on multiple line cards, centralized designs require data to be internally forwarded from those network processors to the push server, presumably on a route processor, which then combines the individual data items into a single consolidated stream. The centralized data collection mechanism can result in a performance bottleneck, especially when large amounts of data are involved.

What is needed is a mechanism that allows for directly publishing from multiple network processors on line cards, without passing them through an additional processing stage for internal consolidation. The proposed UDP-based transport allows for such a distributed data publishing approach.

* Firstly, a UDP approach reduces the burden of maintaining a large amount of active TCP connections at the receiver, notably in cases where it collects data from network processors on line cards from a large amount of networking devices.

* Secondly, as no connection state needs to be maintained, UDP encapsulation can be easily implemented by the hardware of the publication streamer, which will further improve performance.

* Ultimately, such advantages allow for a larger data analysis feature set, as more voluminous, finer grained data sets can be streamed to the receiver.

The transport described in this document can be used for transmitting notification messages over both IPv4 and IPv6.

This document describes the notification mechanism. It is intended to be used in conjunction with [RFC8639], extended by [I-D.ietf-netconf-distributed-notif].

Section 2 describes the control of the proposed transport mechanism. Section 3 details the notification mechanism and message format. Section 4 describes the use of options in the notification message header. Section 5 covers the applicability of the proposed mechanism. Section 6 describes a mechanism to secure the protocol in open networks.

2. Configured Subscription to UDP-Notif

This section describes how the proposed mechanism can be controlled using subscription channels based on NETCONF or RESTCONF.

Following the usual approach of Sub-Notif, configured subscriptions contain the location information of all the receivers, including the IP address and the port number, so that the publisher can actively send UDP-Notif messages to the corresponding receivers.

Note that receivers MAY NOT be already up and running when the configuration of the subscription takes effect on the monitored device. The first message MUST be a separate subscription-started notification to indicate the Receiver that the stream has started flowing. Then, the notifications can be sent immediately without delay. All the subscription state notifications, as defined in [RFC8639], MUST be encapsulated in separate notification messages.

3. UDP-Based Transport

In this section, we specify the UDP-Notif Transport behavior. Section 3.1 describes the general design of the solution. Section 3.2 specifies the UDP-Notif message format. Section 4 describes a generic optional sub TLV format. Section 4.1 uses such options to provide a segmentation solution for large UDP-Notif message payloads. Section 3.3 describes the encoding of the message payload.

3.1. Design Overview

As specified in Sub-Notif, the telemetry data is encapsulated in the NETCONF/RESTCONF notification message, which is then encapsulated and carried using transport protocols such as TLS or HTTP2. This document defines a UDP based transport. Figure 1 illustrates the structure of an UDP-Notif message.

* The Message Header contains information that facilitate the message transmission before deserializing the notification message.

* Notification Message is the encoded content that the publication stream transports. The common encoding methods are listed in Section 3.2. [I-D.ietf-netconf-notification-messages] describes the structure of the Notification Message for single notifications and bundled notifications.

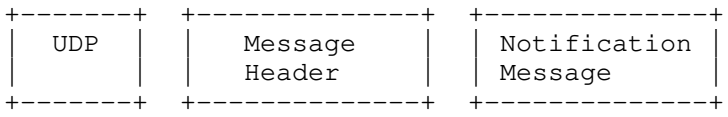


Figure 1: UDP-Notif Message Overview

3.2. Format of the UDP-Notif Message Header

The UDP-Notif Message Header contains information that facilitate the message transmission before deserializing the notification message. The data format is shown in Figure 2.

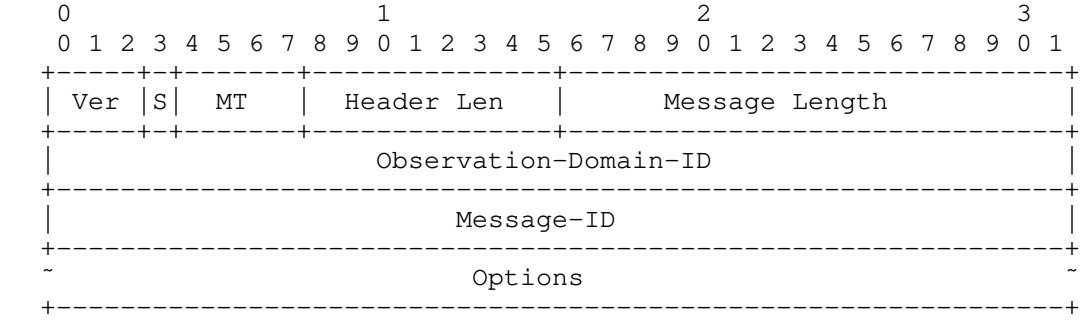


Figure 2: UDP-Notif Message Header Format

The Message Header contains the following field:

* Ver represents the PDU (Protocol Data Unit) encoding version. The initial version value is 0.

* S represents the space of media type specified in the MT field. When S is unset, MT represents the standard media types as defined in this document. When S is set, MT represents a private space to be freely used for non standard encodings.

* MT is a 4 bit identifier to indicate the media type used for the Notification Message. 16 types of encoding can be expressed. When the S bit is unset, the following values apply:

  - 0: Reserved;

  - 1: application/yang-data+json [RFC8040]

  - 2: application/yang-data+xml [RFC8040]

  - 3: application/yang-data+cbor [I-D.ietf-core-yang-cbor]

* Header Len is the length of the message header in octets, including both the fixed header and the options.

* Message Length is the total length of the message within one UDP datagram, measured in octets, including the message header.

* Observation-Domain-ID is a 32-bit identifier of the Observation Domain that led to the production of the notification message, as defined in [I-D.ietf-netconf-notification-messages]. This allows disambiguation of an information source, such as the identification of different line cards sending the notification messages. The source IP address of the UDP datagrams SHOULD NOT be interpreted as the identifier for the host that originated the UDP-Notif message. Indeed, the streamer sending the UDP-Notif message could be a relay for the actual source of data carried within UDP-Notif messages.

* The Message ID is generated continuously by the publisher of UDP-Notif messages. Different subscribers share the same Message ID sequence.

* Options is a variable-length field in the TLV format. When the Header Length is larger than 12 octets, which is the length of the fixed header, Options TLVs follow directly after the fixed message header (i.e., Message ID). The details of the options are described in Section 4.

3.3. Data Encoding

UDP-Notif message data can be encoded in CBOR, XML or JSON format. It is conceivable that additional encodings may be supported in the future. This can be accomplished by augmenting the subscription data model with additional identity statements used to refer to requested encodings.

Private encodings can be supported through the use of the S bit of the header. When the S bit is set, the value of the MT field is left to be defined and agreed upon by the users of the private encoding. An option is defined in Section 4.2 for more verbose encoding descriptions than what can be described with the MT field.

Implementation MAY support multiple encoding methods per subscription. When bundled notifications are supported between the publisher and the receiver, only subscribed notifications with the same encoding can be bundled in a given message.

4. Options

All the options are defined with the following format, illustrated in Figure 3.

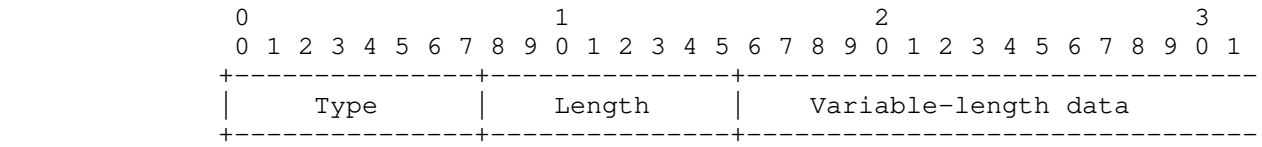


Figure 3: Generic Option Format

* Type: 1 octet describing the option type;

* Length: 1 octet representing the total number of octets in the TLV, including the Type and Length fields;

* Variable-length data: 0 or more octets of TLV Value.

When more than one option are used in the UDP-notif header, options MUST be ordered by the Type value.

4.1. Segmentation Option

The UDP payload length is limited to 65535. Application level headers will make the actual payload shorter. Even though binary encodings such as CBOR may not require more space than what is left, more voluminous encodings such as JSON and XML may suffer from this size limitation. Although IPv4 and IPv6 publishers can fragment outgoing packets exceeding their Maximum Transmission Unit(MTU), fragmented IP packets may not be desired for operational and performance reasons.

Consequently, implementations of the mechanism SHOULD provide a configurable max-segment-size option to control the maximum size of a payload.

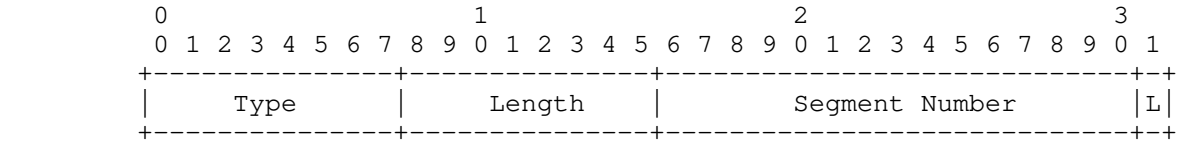


Figure 4: Segmentation Option Format

The Segmentation Option is to be included when the message content is segmented into multiple pieces. Different segments of one message share the same Message ID. An illustration is provided in Figure 4. The fields of this TLV are:

* Type: Generic option field which indicates a Segmentation Option. The Type value is to be assigned TBD1.

* Length: Generic option field which indicates the length of this option. It is a fixed value of 4 octets for the Segmentation Option.

* Segment Number: 15-bit value indicating the sequence number of the current segment. The first segment of a segmented message has a Segment Number value of 0.

* L: is a flag to indicate whether the current segment is the last one of the message. When 0 is set, the current segment is not the last one. When 1 is set, the current segment is the last one, meaning that the total number of segments used to transport this message is the value of the current Segment Number + 1.

An implementation of this specification MUST NOT rely on IP fragmentation by default to carry large messages. An implementation of this specification MUST either restrict the size of individual messages carried over this protocol, or support the segmentation option.

When a message has multiple options and is segmented using the described mechanism, all the options MUST be present on the first segment ordered by the options Type. The rest of segmented messages MAY include all the options ordered by options type.

4.2. Private Encoding Option

The space to describe private encodings in the MT field of the UDP-Notif header being limited, an option is provided to describe custom encodings. The fields of this option are as follows.

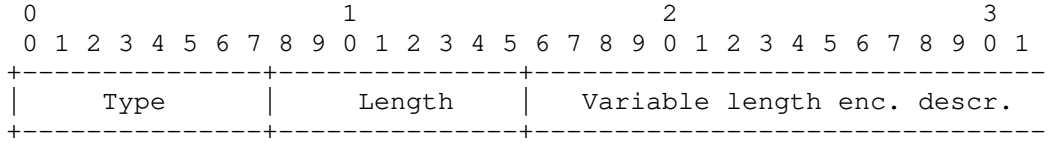


Figure 5: Private Encoding Option Format

* Type: Generic option field which indicates a Private Encoding Option. The Type value is to be assigned TBD2.

* Length: Generic option field which indicates the length of this option. It is a variable value.

* Enc. Descr: The description of the private encoding used for this message. The values to be used for such private encodings is left to be defined by the users of private encodings.

This option SHOULD only be used when the S bit of the header is set, as providing a private encoding description for standard encodings is meaningless.

5. Applicability

In this section, we provide an applicability statement for the proposed mechanism, following the recommendations of [RFC8085].

The proposed mechanism falls in the category of UDP applications "designed for use within the network of a single network operator or on networks of an adjacent set of cooperating network operators, to

be deployed in controlled environments". Implementations of the proposed mechanism SHOULD thus follow the recommendations in place for such specific applications. In the following, we discuss recommendations on congestion control, message size guidelines, reliability considerations and security considerations.

5.1. Congestion Control

The proposed application falls into the category of applications performing transfer of large amounts of data. It is expected that the operator using the solution configures QoS on its related flows. As per [RFC8085], such applications MAY choose not to implement any form of congestion control, but follow the following principles.

It is NOT RECOMMENDED to use the proposed mechanism over congestion-sensitive network paths. The only environments where UDP-Notif is expected to be used are managed networks. The deployments require that the network path has been explicitly provisioned to handle the traffic through traffic engineering mechanisms, such as rate limiting or capacity reservations.

Implementation of the proposal SHOULD NOT push unlimited amounts of traffic by default, and SHOULD require the users to explicitly configure such a mode of operation.

Burst mitigation through packet pacing is RECOMMENDED. Disabling burst mitigation SHOULD require the users to explicitly configure such a mode of operation.

Applications SHOULD monitor packet losses and provide means to the user for retrieving information on such losses. The UDP-Notif Message ID can be used to deduce congestion based on packet loss detection. Hence the receiver can notify the device to use a lower streaming rate. The interaction to control the streaming rate on the device is out of the scope of this document.

5.2. Message Size

[RFC8085] recommends not to rely on IP fragmentation for messages whose size result in IP packets exceeding the MTU along the path. The segmentation option of the current specification permits segmentation of the UDP Notif message content without relying on IP fragmentation. Implementation of the current specification SHOULD allow for the configuration of the MTU.

5.3. Reliability

The target application for UDP-Notif is the collection of data-plane information. The lack of reliability of the data streaming mechanism is thus considered acceptable as the mechanism is to be used in controlled environments, mitigating the risk of information loss, while allowing for publication of very large amounts of data. Moreover, in this context, sporadic events when incomplete data collection is provided is not critical for the proper management of the network, as information collected for the devices through the means of the proposed mechanism is to be often refreshed.

A receiver implementation for this protocol SHOULD deal with potential loss of packets carrying a part of segmented payload, by discarding packets that were received, but cannot be re-assembled as a complete message within a given amount of time. This time SHOULD be configurable.

5.4. Security Considerations

[RFC8085] states that "UDP applications that need to protect their communications againts eavesdropping, tampering, or message forgery SHOULD employ end-to-end security services provided by other IETF protocols". As mentioned above, the proposed mechanism is designed to be used in controlled environments and thus, a security layer is unrequired. Nevertheless, a DTLS layer SHOULD be implemented in open or unsecured networks. A DTLS layered implementation is presented in Section 6.

6. Secured layer for UDP-notif

In open or unsecured networks, UDP-notif messages SHOULD be secured or encrypted. In this section, a mechanism using DTLS 1.3 to secure UDP-notif protocol is presented. The following sections defines the requirements for the implementation of the secured layer of DTLS for UDP-notif. No DTLS 1.3 extensions are defined nor needed.

The DTLS 1.3 protocol [I-D.draft-ietf-tls-dtls13] is designed to meet the requirements of applications that need to secure datagram transport.

DTLS can be used as a secure transport to counter all the primary threats to UDP-notif:

* Confidentiality to counter disclosure of the message contents.

* Integrity checking to counter modifications to a message on a hop-by-hop basis.

* Server or mutual authentication to counter masquerade.

In addition, DTLS also provides:

* A cookie exchange mechanism during handshake to counter Denial of Service attacks.

* A sequence number in the header to counter replay attacks.

Even though this security layer is unrequired, DTLS 1.3 SHOULD be implemented on unsecured networks to achieve privacy.

6.1. Transport

As shown in Figure 6, the DTLS is layered next to the UDP transport providing reusable security and authentication functions over UDP. No DTLS extension is required to enable UDP-notif messages over DTLS.

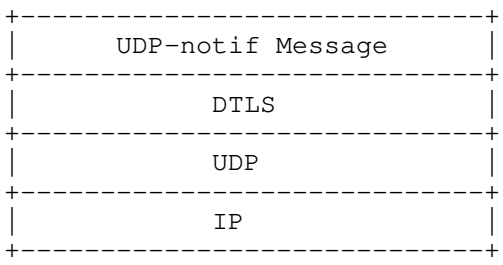


Figure 6: Protocol Stack for DTLS secured UDP-notif

The application implementer will map a unique combination of the remote address, remote port number, local address, and local port number to a session.

Each UDP-notif message is delivered by the DTLS record protocol, which assigns a sequence number to each DTLS record. Although the DTLS implementer may adopt a queue mechanism to resolve reordering, it may not assure that all the messages are delivered in order when mapping on the UDP transport.

Since UDP is an unreliable transport, with DTLS, an originator or a relay may not realize that a collector has gone down or lost its DTLS connection state, so messages may be lost.

The DTLS record has its own sequence number, encryption and decryption will be done by the DTLS layer, so that the UDP-notif Message layer is not impacted by the use of DTLS.

6.2. Port Assignment

When this security layer is used, the Publisher MUST always be a DTLS client, and the Receiver MUST always be a DTLS server. The Receivers MUST support accepting UDP-notif Messages on the specified UDP port, but MAY be configurable to listen on a different port. The Publisher MUST support sending UDP-notif messages to the specified UDP port, but MAY be configurable to send messages to a different port. The Publisher MAY use any source UDP port for transmitting messages.

6.3. Session lifecycle

6.3.1. DTLS Session Initiation

The Publisher initiates a DTLS connection by sending a DTLS ClientHello to the Receiver. Implementations MAY support the denial of service countermeasures defined by DTLS 1.3. When these countermeasures are used, the Receiver responds with a DTLS HelloRetryRequest containing a stateless cookie. The Publisher MUST send a new DTLS ClientHello message containing the received cookie, which initiates the DTLS handshake.

When DTLS is implemented, the Publisher MUST NOT send any UDP-notif messages before the DTLS handshake has successfully completed.

Implementations of this security layer MUST support DTLS 1.3 [I-D.draft-ietf-tls-dtls13] and MUST support the mandatory to implement cipher suite TLS_AES_128_GCM_SHA256 and SHOULD implement TLS_AES_256_GCM_SHA384 and TLS_CHACHA20_POLY1305_SHA256 cipher suites, as specified in TLS 1.3 [RFC8446]. If additional cipher suites are supported, then implementations MUST NOT negotiate a cipher suite that employs NULL integrity or authentication algorithms.

Where privacy is REQUIRED, then implementations must either negotiate a cipher suite that employs a non-NULL encryption algorithm or otherwise achieve privacy by other means, such as a physically secured network.

6.3.2. Publish Data

When DTLS is used, all UDP-notif messages MUST be published as DTLS "application_data". It is possible that multiple UDP-notif messages are contained in one DTLS record, or that a publication message is transferred in multiple DTLS records. The application data is defined with the following ABNF [RFC5234] expression:

```
APPLICATION-DATA = 1*UDP-NOTIF-FRAME
```

```
UDP-NOTIF-FRAME = MSG-LEN SP UDP-NOTIF-MSG

MSG-LEN = NONZERO-DIGIT *DIGIT

SP = %d32

NONZERO-DIGIT = %d49-57

DIGIT = %d48 / NONZERO-DIGIT
```

UDP-NOTIF-MSG is defined in Section 3.

The Publisher SHOULD attempt to avoid IP fragmentation by using the Segmentation Option in the UDP-notif message.

6.3.3. Session termination

A Publisher MUST close the associated DTLS connection if the connection is not expected to deliver any UDP-notif Messages later. It MUST send a DTLS close_notify alert before closing the connection. A Publisher (DTLS client) MAY choose to not wait for the Receiver's close_notify alert and simply close the DTLS connection. Once the Receiver gets a close_notify from the Publisher, it MUST reply with a close_notify.

When no data is received from a DTLS connection for a long time, the Receiver MAY close the connection. Implementations SHOULD set the timeout value to 10 minutes but application specific profiles MAY recommend shorter or longer values. The Receiver (DTLS server) MUST attempt to initiate an exchange of close_notify alerts with the Publisher before closing the connection. Receivers that are unprepared to receive any more data MAY close the connection after sending the close_notify alert.

Although closure alerts are a component of TLS and so of DTLS, they, like all alerts, are not retransmitted by DTLS and so may be lost over an unreliable network.

7. A YANG Data Model for Management of UDP-Notif

The YANG model defined in Section 8 has two leaves augmented into one place of Sub-Notif [RFC8639], plus one identity.

```
    module: ietf-udp-subscribed-notifications
     augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
       +--rw address   inet:ip-address
       +--rw port      inet:port-number
       +--rw enable-segmentation?  boolean
       +--rw max-segmentation-size?  uint32
```

8. YANG Module

```
   <CODE BEGINS> file "ietf-udp-notif@2020-10-18.yang"
   module ietf-udp-notif {
     yang-version 1.1;
     namespace
       "urn:ietf:params:xml:ns:yang:ietf-udp-notif";
     prefix un;
     import ietf-subscribed-notifications {
       prefix sn;
       reference
         "RFC 8639: Subscription to YANG Notifications";
     }
     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     organization "IETF NETCONF (Network Configuration) Working Group";
     contact
       "WG Web:   <http:/tools.ietf.org/wg/netconf/>
        WG List:  <mailto:netconf@ietf.org>

        Authors:  Guangying Zheng
                  <mailto:zhengguangying@huawei.com>
                  Tianran Zhou
                  <mailto:zhoutianran@huawei.com>
                  Thomas Graf
                  <mailto:thomas.graf@swisscom.com>
                  Pierre Francois
                  <mailto:pierre.francois@insa-lyon.fr>
                  Paolo Lucente
                  <mailto:paolo@ntt.net>";

     description
       "Defines UDP-Notif as a supported transport for subscribed
       event notifications.

       Copyright (c) 2018 IETF Trust and the persons identified as authors
```

```
      of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or without
      modification, is permitted pursuant to, and subject to the license
      terms contained in, the Simplified BSD License set forth in Section
      4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
      (https://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see the RFC
      itself for full legal notices.";

    revision 2021-10-18 {
      description
      "Slight change to the name of two parameters.";
      reference
      "RFC XXXX: UDP-based Transport for Configured Subscriptions";
    }

   /*
    * FEATURES
    */
    feature encode-cbor {
      description
        "This feature indicates that CBOR encoding of notification
         messages is supported.";
    }

   /*
    * IDENTITIES
    */
    identity udp-notif {
      base sn:transport;
      description
     "UDP-Notif is used as transport for notification messages
        and state change notifications.";
    }

    identity encode-cbor {
      base sn:encoding;
      description
        "Encode data using CBOR as described in RFC XXX.";
      reference
        "RFC XXX: draft-ietf-core-yang-cbor-18, CBOR Encoding of
          Data Modeled with YANG";
    }

    grouping target-receiver {
      description
```

```
        "Provides a reusable description of a UDP-Notif target
        receiver.";

      leaf address {
        type inet:ip-address;
        mandatory true;
        description
          "IP address of target UDP-Notif receiver, which can be an
          IPv4 address or an IPV6 address.";
      }

      leaf port {
      type inet:port-number;
      description
        "Port number of target UDP-Notif receiver, if not specified,
        the system should use default port number.";
      }

      leaf enable-segmentation {
        type boolean;
        default false;
        description
          "The switch for the segmentation feature. When disabled, the
          publisher will not allow fragment for a very large data";
      }

      leaf max-segmentation-size {
      when "../enable-segmentation = 'true'";
      type uint32;
      description "UDP-Notif provides a configurable
        max-segmentation-size to control the size of each message.";
      }
    }

    augment "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
      when "derived-from(../../../transport, 'un:udp-notif')";
      description
        "This augmentation allows UDP-Notif specific parameters to be
         exposed for a subscription.";

      uses target-receiver;
    }
  }
  <CODE ENDS>
```

9. IANA Considerations

This document is creating 2 registries called "UDP-notif media types" and "UDP-notif option types" under the new heading "UDP-notif protocol". The registration procedure is made using the Standards Action process defined in [RFC8126].

The first requested registry is the following:

Registry Name: UDP-notif media types
Registry Category: UDP-notif protocol.
Registration Procedure: Standard Action as defined in RFC8126
Maximum value: 15

These are the initial registrations for "UDP-notif media types":

Value: 0
Description: Reserved
Reference: this document

Value: 1
Description: media type application/yang-data+json
Reference: <xref target="RFC8040"/>

Value: 2
Description: media type application/yang-data+xml
Reference: <xref target="RFC8040"/>

Value: 3
Description: media type application/yang-data+cbor
Reference: <xref target="I-D.ietf-core-yang-cbor"/>

The second requested registry is the following:

Registry Name: UDP-notif option types
Registry Category: UDP-notif protocol.
Registration Procedure: Standard Action as defined in RFC8126
Maximum value: 255

These are the initial registrations for "UDP-notif options types":

Value: 0
Description: Reserved
Reference: this document

Value: TBD1 (suggested value: 1)
Description: Segmentation Option
Reference: this document

Value: TBD2 (suggested value: 2)
Description: Private Encoding Option
Reference: this document

IANA is also requested to assign a new URI from the IETF XML Registry [RFC3688]. The following URI is suggested:

URI: urn:ietf:params:xml:ns:yang:ietf-udp-notif
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document also requests a new YANG module name in the YANG Module Names registry [RFC7950] with the following suggestion:

name: ietf-udp-notif
namespace: urn:ietf:params:xml:ns:yang:ietf-udp-notif
prefix: un
reference: RFC XXXX

10. Acknowledgements

The authors of this documents would like to thank Alexander Clemm, Eric Voit, Huiyang Yang, Kent Watsen, Mahesh Jethanandani, Stephane Frenot, Timothy Carey, Tim Jenkins, Yunan Gu and Marco Tollini for their constructive suggestions for improving this document.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8640] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Dynamic Subscription to YANG Events and Datastores over NETCONF", RFC 8640, DOI 10.17487/RFC8640, September 2019, <https://www.rfc-editor.org/info/rfc8640>.

[RFC8650] Voit, E., Rahman, R., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Dynamic Subscription to YANG Events and Datastores over RESTCONF", RFC 8650, DOI 10.17487/RFC8650, November 2019, <https://www.rfc-editor.org/info/rfc8650>.

11.2. Informative References

[I-D.draft-ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls13-43>.

[I-D.ietf-core-yang-cbor]
Veillette, M., Petrov, I., Pelov, A., Bormann, C., and M. Richardson, "CBOR Encoding of Data Modeled with YANG", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-18, 19 December 2021, <https://www.ietf.org/archive/id/draft-ietf-core-yang-cbor-18.txt>.

[I-D.ietf-netconf-distributed-notif]
Zhou, T., Zheng, G., Voit, E., Graf, T., and P. Francois, "Subscription to Distributed Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-distributed-notif-02, May 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-distributed-notif-02>.

[I-D.ietf-netconf-https-notif]
Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for YANG Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-https-notif-09, 24 October 2021, <https://www.ietf.org/archive/id/draft-ietf-netconf-https-notif-09.txt>.

[I-D.ietf-netconf-notification-messages]
Voit, E., Jenkins, T., Birkholz, H., Bierman, A., and A. Clemm, "Notification Message Headers and Bundles", Work in Progress, Internet-Draft, draft-ietf-netconf-notification-messages-08, 17 November 2019, <https://www.ietf.org/archive/id/draft-ietf-netconf-notification-messages-08.txt>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

Authors' Addresses

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing
Jiangsu,
China
Email: zhengguangying@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China
Email: zhoutianran@huawei.com

Thomas Graf
Swisscom
Binzring 17
CH- Zuerich 8045
Switzerland
Email: thomas.graf@swisscom.com

Pierre Francois
INSA-Lyon
Lyon
France
Email: pierre.francois@insa-lyon.fr

Alex Huang Feng
INSA-Lyon
Lyon
France
Email: alex.huang-feng@insa-lyon.fr

Paolo Lucente
NTT
Siriusdreef 70-72
Hoofddorp, WT 2132
Netherlands
Email: paolo@ntt.net

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: 25 April 2022

J. Lindblad
Cisco Systems

22 October 2021

Transaction ID Mechanism for NETCONF
draft-lindblad-netconf-transaction-id-01

Abstract

NETCONF clients and servers often need to have a synchronized view of the server's configuration data stores. The volume of configuration data in a server may be very large, while data store changes typically are small when observed at typical client resynchronization intervals.

Rereading the entire data store and analyzing the response for changes is an inefficient mechanism for synchronization. This document specifies an extension to NETCONF that allows clients and servers to keep synchronized with a much smaller data exchange and without any need for servers to store information about the clients.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/janlindblad/netconf-transaction-id.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 April 2022.

Table of Contents

1. Introduction 2
2. Conventions and Definitions 3
3. NETCONF Transaction id Extension 3
3.1. General Principles 4
3.2. Conditional Transactions 5
3.3. Other NETCONF Operations 6
4. ETag Transaction id Mechanism 6
4.1. ETag attribute 6
4.2. Configuration Retreival 7
4.2.1. Initial Configuration Response 7
4.2.2. Configuration Response Pruning 9
4.3. Configuration Update 11
4.3.1. Conditional Configuration Update 14
4.4. ETags with Other NETCONF Operations 16
5. YANG Modules 17
6. Security Considerations 20
7. IANA Considerations 20
8. Changes 21
8.1. Major changes in -01 since -00 21
9. Normative References 22
Acknowledgments 22
Author's Address 22

1. Introduction

When a NETCONF client connects with a NETCONF server, a frequently occurring use case is for the client to find out if the configuration has changed since it was last connected. Such changes could occur for example if another NETCONF client has made changes, or another system or operator made changes through other means than NETCONF.

One way of detecting a change for a client would be to retrieve the entire configuration from the server, then compare the result with a previously stored copy at the client side. This approach is not popular with most NETCONF users, however, since it would often be very expensive in terms of communications and computation cost.

Furthermore, even if the configuration is reported to be unchanged, that will not guarantee that the configuration remains unchanged when a client sends a subsequent change request, a few moments later.

Evidence of a transaction id feature being demanded by clients is that several server implementors have built proprietary and mutually incompatible mechanisms for obtaining a transaction id from a NETCONF server.

RESTCONF, RFC 8040 (https://tools.ietf.org/html/rfc8040), defines a mechanism for detecting changes in configuration subtrees based on Entity-tags (ETags). In conjunction with this, RESTCONF provides a way to make configuration changes conditional on the server confiuguration being untouched by others. This mechanism leverages RFC 7232 (https://tools.ietf.org/html/rfc7232) "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests".

This document defines similar functionality for NETCONF, RFC 6241 (https://tools.ietf.org/html/rfc6241).

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. NETCONF Transaction id Extension

This document describes a NETCONF extension which modifies the behavior of get-config, get-data, edit-config, edit-data, discard-changes, copy-config, delete-config and commit such that clients are able to conditionally retrieve and update the configuration in a NETCONF server. NETCONF servers that support this extension MUST announce the capability "urn:ietf:params:netconf:capability:txid:1.0".

Several low level mechanisms could be defined to fulfill the requirements for efficient client-server transaction id synchronization. This document defines only one mechanism, but additional mechanisms could be added in future versions of this document, or in separate documents.

The common use cases for such mecahnisms are briefly discussed here.

Initial configuration retrieval When the client initially connects to a server, it may be interested to acquire a current view of (parts of) the server's configuration.
In order to be able to efficiently detect changes later, it may also be interested to store meta level transaction id information about subtrees of the configuration.

Subsequent configuration retrieval When a client needs to reread (parts of) the server's configuration, it may be interested to leverage the transaction id meta data it has stored by requesting the server to prune the response so that it does not repeat configuration data that the client is already aware of.

Configuration update with transaction id return When a client issues a transaction towards a server, it may be interested to also learn the new transaction id meta data the server has stored for the updated parts of the configuration.

Configuration update with transaction id specification When a client issues a transaction towards a server, it may be interested to also specify the new transaction id meta data that the server stores for the updated parts of the configuration.

Conditional configuration update When a client issues a transaction towards a server, it may specify transaction id data for the transaction in order to allow the server to verify that the client is up to date with any changes in the parts of the configuration that it is concerned with. If the transaction id information in the server is different than the client expected, the server rejects the transaction with a specific error message.

3.1. General Principles

All transaction id mechanisms SHALL maintain a transaction id value for each configuration datastore supported by the server. Some transaction id mechanisms will also maintain transaction id values for elements deeper in the YANG data tree. The elements for which the server maintains transaction ids are collectively referred to as the "versioned elements".

The server returning transaction id values for the versioned elements MUST ensure the transaction id values are changed every time there has been a configuration change at or below the element associated with the value. This means any update of a config true element will result in a new transaction id value for all ancestor versioned elements, up to and including the datastore root itself.

This also means a server MUST update the transaction id value for any elements that change as a result of a configuration change, regardless of source, even if the changed elements are not explicitly part of the change payload. An example of this is dependent data under YANG RFC 7950 (https://tools.ietf.org/html/rfc7950) when- or choice-statements.

The server MUST NOT change the transaction id value of a versioned element unless a child element of that element has been changed. The server MUST NOT change any transaction id values due to changes in config false data.

3.2. Conditional Transactions

Conditional transactions are useful when a client is interested to make a configuration change, being sure that the server configuration has not changed since the client last inspected it.

By supplying the latest transaction id values known to the client in its change requests (edit-config etc.), it can request the server to reject the transaction in case any relevant changes have occurred at the server that the client is not yet aware of.

This allows a client to reliably compute and send confiuguration changes to a server without either acquiring a global datastore lock for a potentially extended period of time, or risk that a change from another client disrupts the intent in the time window between a read (get-config etc.) and write (edit-config etc.) operation.

If the server rejects the transaction because the configuration transaction id value differs from the client's expectation, the server MUST return an rpc-error with the following values:

```
error-tag:      operation-failed
error-type:     protocol
error-severity: error
```

Additionally, the error-info tag SHOULD contain an sx:structure containing relevant details about the mismatching transaction ids.

3.3. Other NETCONF Operations

discard-changes The discard-changes operation resets the candidate datastore to the contents of the running datastore. The server MUST ensure the transaction id values in the candidate datastore get the same values as in the running datastore when this operation runs.

copy-config The copy-config operation can be used to copy contents between datastores. The server MUST ensure the transaction id values retain the same values as in the soruce datastore.

If copy-config is used to copy from a file, URL or other source that is not a datastore, the server MUST ensure the transaction id values are changed.

delete-config The server MUST ensure the datastore transaction id value is changed.

commit At commit, with regards to the transaction id values, the server MUST treat the contents of the candidate datastore as if any transaction id value provided by the client when updating the candidate was provided in a single edit-config towards the running datastore. If the transaction is rejected due to transaction id value mismatch, an rpc-error as described in section Conditional Transactions (Section 3.2) MUST be sent.

4. ETag Transaction id Mechanism

4.1. ETag attribute

Central to the ETag configuration retrieval and update mechanism described in the following sections is a meta data XML attribute called "etag". The etag attribute is defined in the namespace "urn:ietf:params:xml:ns:netconf:txid:1.0".

Servers MUST maintain a top-level etag value for each configuration datastore they implement. Servers SHOULD maintain etag values for YANG containers that hold configuration for different subsystems. Servers MAY maintain etag values for any YANG container or list element they implement.

The etag attribute values are opaque UTF-8 strings chosen freely, except that the etag string must not contain space, backslash or double quotes. The point of this restriction is to make it easy to reuse implementations that adhere to section 2.3.1 in RFC 7232 (https://tools.ietf.org/html/rfc7232). The probability SHOULD be made very low that an etag value that has been used historically by a server is used again by that server.

The detailed rules for when to update the etag value are described in section Configuration Update (Section 4.3). These rules are chosen to be consistent with the ETag mechanism in RESTCONF, RFC 8040 (https://tools.ietf.org/html/rfc8040), specifically sections 3.4.1.2, 3.4.1.3 and 3.5.2.

4.2. Configuration Retreival

Clients MAY request the server to return etag attribute values in the response by adding one or more etag attributes in get-config or get-data requests.

The etag attribute may be added directly on the get-config or get-data requests, in which case it pertains to the entire datastore. A client MAY also add etag attributes to zero or more individual elements in the get-config or get-data filter, in which case it pertains to the subtree rooted at that element.

For each element that the client requests etag attributes, the server MUST return etags for all versioned elements at or below that point that are part of the server's respone. ETags are returned as attributes on the element they pertain to. The datastore root etag value is returned on the top-level data tag in the response.

If the client is requesting an etag value for an element that is not among the server's versioned elements, then the server MUST return the etag attribute on the closest ancestor that is a versioned element, and all children of that ancestor. The datastore root is always a versioned element.

4.2.1. Initial Configuration Response

When the client adds etag attributes to a get-config or get-data request, it should specify the last known etag values it has seen for the elements it is asking about. Initially, the client will not know any etag value and should use "?".

To retrieve etag attributes across the entire NETCONF server configuration, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config txid:etag="?"/>
</rpc>
```

To retrieve etag attributes for a specific interface using an xpath filter, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="xpath"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
      select=
        "/if:interfaces/if:interface[if:name='GigabitEthernet-0/0']"
      txid:etag="?"/>
  </get-config>
</rpc>
```

To retrieve etag attributes for "ietf-interfaces", but not for "nacm", a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        txid:etag="?"/>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
    </filter>
  </get-config>
</rpc>
```

When a NETCONF server receives a get-config or get-data request containing txid:etag attributes with the value "?", it MUST return etag attributes for all versioned elements below this point included in the reply.

If the server considers the container "interfaces" and the list "interface" elements to be versioned elements, the server's response to the request above might look like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="def88884321">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface txid:etag="abc12345678">
        <name>GigabitEthernet-0/1</name>
        <description>Upward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
      <groups>
        <group>
          <name>admin</name>
          <user-name>sakura</user-name>
          <user-name>joe</user-name>
        </group>
      </groups>
    </nacm>
  </data>
</rpc>
```

4.2.2. Configuration Response Pruning

A NETCONF client that already knows some etag values MAY request that the configuration retrieval request is pruned with respect to the client's prior knowledge.

To retrieve only changes for "ietf-interfaces" that do not have the last known etag value "abc12345678", but include the entire configuration for "nacm", regardless of etags, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        txid:etag="abc12345678"/>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
    </filter>
  </get-config>
</rpc>
```

When a NETCONF server receives a get-config or get-data request containing an element with a client specified etag attribute, there are several different cases:

* The element is not a versioned element, i.e. the server does not maintain an etag value for this element. In this case, the server MUST look up the closest ancestor that is a versioned element, and proceed as if the client had specified the etag value for that element.

* The element is a versioned element, and the client specified etag attribute value is different than the server's etag value for this element. In this case the server MUST return the contents as it would otherwise have done, adding the etag attributes of all child versioned elements to the response. In case the client has specified etag attributes for some child elements, then these cases MUST be re-evaluated for those elements.

* The element is a versioned element, and the client specified etag attribute value matches the server's etag value. In this case the server MUST return the element decorated with an etag attribute with the value "=", and child elements pruned.

For list elements, pruning child elements means that key elements MUST be included in the response, and other child elements MUST NOT be included. For containers, child elements MUST NOT be included.

For example, assuming the NETCONF server configuration is the same as in the previous rpc-reply example, the server's response to request above might look like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="def88884321">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface txid:etag="=">
        <name>GigabitEthernet-0/1</name>
      </interface>
    </interfaces>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
      <groups>
        <group>
          <name>admin</name>
          <user-name>sakura</user-name>
          <user-name>joe</user-name>
        </group>
      </groups>
    </nacm>
  </data>
</rpc>
```

4.3. Configuration Update

Whenever the configuration on a server changes for any reason, the server MUST update the etag value for all versioned elements that have children that changed.

If the change is due to a NETCONF client edit-config or edit-data request that includes the ietf-netconf-txid:with-etag presence container, the server MUST return the etag value of the targeted datastore as an attribute on the XML ok tag in the rpc-reply.

The server MUST NOT change the etag value of a versioned element unless a child element of that element has been changed. The server MUST NOT change any etag values due to changes in config false data.

How the server selects a new etag value to use for the changed elements is described in section ETag attribute (Section 4.1).

For example, if a client wishes to update the interface description for interface "GigabitEthernet-0/1" to "Downward Interface", it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
               xmlns:ietf-netconf-txid=
                "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <ietf-netconf-txid:with-etag/>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet-0/1</name>
          <description>Downward Interface</description>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

The server would update the description leaf in the candidate datastore, and return an rpc-reply as follows:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="ghi55550101"/>
</rpc-reply>
```

A subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="ghi55550101">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="ghi55550101">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface txid:etag="ghi55550101">
        <name>GigabitEthernet-0/1</name>
        <description>Downward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

In case the server at this point received a configuration change from another source, such as a CLI operator, adding an MTU value for the interface "GigabitEthernet-0/0", a subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="cli22223333">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="cli22223333">
      <interface txid:etag="cli22223333">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <mtu>768</mtu>
      </interface>
      <interface txid:etag="ghi55550101">
        <name>GigabitEthernet-0/1</name>
        <description>Downward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

4.3.1. Conditional Configuration Update

When a NETCONF client sends an edit-config or edit-data request to a NETCONF server that implements this specification, the client MAY specify expected etag values on the versioned elements touched by the transaction.

If such an etag value differs from the etag value stored on the server, the server MUST reject the transaction and return an rpc-error as specified in section Conditional Transactions (Section 3.2).

Additionally, the error-info tag MUST contain an sx:structure etag-value-mismatch-error-info as defined in the module ietf-netconf-txid, with mismatch-path set to the instance identifier value identifying one of the versioned elements that had an etag value mismatch, and mismatch-etag-value set to the server's current value of the etag attribute for that versioned element.

For example, if a client wishes to delete the interface "GigabitEthernet-0/1" if and only if its configuration has not been altered since this client last synchronized its configuration with the server (at which point it received the etag "ghi55550101"), regardless of any possible changes to other interfaces, it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0"
     xmlns:ietf-netconf-txid=
       "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <ietf-netconf-txid:with-etag/>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface nc:operation="delete"
                   txid:etag="ghi55550101">
          <name>GigabitEthernet-0/1</name>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

If interface "GigabitEthernet-0/1" has the etag value "ghi55550101", as expected by the client, the transaction goes through, and the server responds something like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="xyz77775511"/>
</rpc-reply>
```

A subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="xyz77775511">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="xyz77775511">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

In case interface "GigabitEthernet-0/1" did not have the expected etag value "ghi55550101", the server rejects the transaction, and might send:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
           xmlns:ietf-netconf-txid=
             "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
           message-id="1">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <ietf-netconf-txid:etag-value-mismatch-error-info>
        <ietf-netconf-txid:mismatch-path>
          /if:interfaces/if:interface[if:name="GigabitEthernet-0/0"]
        </ietf-netconf-txid:mismatch-path>
        <ietf-netconf-txid:mismatch-etag-value>
          cli22223333
        </ietf-netconf-txid:mismatch-etag-value>
      </ietf-netconf-txid:etag-value-mismatch-error-info>
    </error-info>
  </rpc-error>
</rpc-reply>
```

4.4. ETags with Other NETCONF Operations

The following NETCONF Operations also need some special considerations.

discard-changes The server MUST ensure the etag attributes in the

candidate datastore get the same values as in the running datastore when this operation runs.

copy-config The server MUST ensure the etag attributes retain the same values as in the soruce datastore.

If copy-config is used to copy from a source that is not a datastore, the server MUST ensure etags are given new values.

delete-config The server MUST ensure the datastore etag is given a new value.

commit At commit, with regards to the etag values, the server MUST treat the contents of the candidate datastore as if any etag attributes provided by the client were provided in a single edit-config towards the running datastore. If the commit is rejected due to etag mismatch, the rpc-error message specified in section Conditional Configuration Update (Section 4.3.1) MUST be sent.

The client MAY request that the new etag value is returned as an attribute on the ok response for a successful commit. The client requests this by adding with-etag to the commit operation.

For example, a client might send:

```
<rpc message-id="1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    xmlns:ietf-netconf-txid=
      "urn:ietf:params:xml:ns:yang:ietf-netconf-txid"
  <commit>
    <ietf-netconf-txid:with-etag/>
  </commit>
</rpc>
```

Assuming the server accepted the transaction, it might respond:

```
<rpc-reply message-id="1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="ghi55550101"/>
</rpc-reply>
```

5. YANG Modules

```
   module ietf-netconf-txid {
     yang-version 1.1;
     namespace
       'urn:ietf:params:xml:ns:yang:ietf-netconf-txid';
     prefix ietf-netconf-txid;

     import ietf-netconf {
       prefix nc;
     }

     import ietf-netconf-nmda {
       prefix ncds;
     }

     import ietf-yang-structure-ext {
       prefix sx;
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netconf/>
        WG List:  <netconf@ietf.org>

        Author:   Jan Lindblad
                  <mailto:jlindbla@cisco.com>";

     description
       "NETCONF Transaction ID aware operations for NMDA.

        Copyright (c) 2021 IETF Trust and the persons identified as
        the document authors.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2021-11-01 {
       description
         "Initial revision";
       reference
```

```
        "RFC XXXX: Xxxxxxxxx";
    }

    typedef etag-t {
      type string {
        pattern ".* .*" {
          modifier invert-match;
        }
        pattern ".*\".*" {
          modifier invert-match;
        }
        pattern ".*\\.*" {
          modifier invert-match;
        }
      }
      description
        "Unique Entity-tag value representing a specific transaction.
         Could be any string that does not contain spaces, double
         quotes or backslash.  The values '?' and '=' have special
         meaning.";
    }

    grouping transaction-id-grouping {
      container with-etag {
        presence
          "Indicates that the client requests the server to include a
           txid:etag transaction id in the rpc-reply";
      }
      description
        "Grouping for transaction id mechanisms, to be augmented into
         rpcs that modify configuration data stores.";
    }

    augment /nc:edit-config/nc:input {
      uses transaction-id-grouping;
      description
        "Injects the transaction id mechanisms into the
        edit-config operation";
    }

    augment /nc:commit/nc:input {
      uses transaction-id-grouping;
      description
        "Injects the transaction id mechanisms into the
        commit operation";
    }

    augment /ncds:edit-data/ncds:input {
```

```
      uses transaction-id-grouping;
      description
        "Injects the transaction id mechanisms into the
        edit-data operation";

    sx:structure etag-value-mismatch-error-info {
      container etag-value-mismatch-error-info {
        description
           "This error is returned by a NETCONF server when a client
            sends a configuration change request, with the additonal
            condition that the server aborts the transaction if the
            server's configuration has changed from what the client
            expects, and the configuration is found not to actually
            not match the client's expectation.";
        leaf mismatch-path {
          type instance-identifier;
          description
            "Indicates the YANG path to the element with a mismatching
             etag value.";
        }
        leaf mismatch-etag-value {
          type etag-t;
          description
            "Indicates server's value of the etag attribute for one
             mismatching element.";
        }
      }
    }
  }
```

6. Security Considerations

TODO Security

7. IANA Considerations

This document registers the following capability identifier URN in the 'Network Configuration Protocol (NETCONF) Capability URNs' registry:

```
  urn:ietf:params:netconf:capability:txid:1.0
```

This document registers two XML namespace URNs in the 'IETF XML registry', following the format defined in RFC 3688 (https://tools.ietf.org/html/rfc3688).

URI: urn:ietf:params:xml:ns:netconf:txid:1.0

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-txid

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URIs are XML namespaces.

This document registers one module name in the 'YANG Module Names' registry, defined in RFC 6020 (https://tools.ietf.org/html/rfc6020).

name: ietf-netconf-txid

prefix: ietf-netconf-txid

namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-txid

RFC: XXXX

8. Changes

8.1. Major changes in -01 since -00

* Updated the text on numerous points in order to answer questions that appeared on the mailing list.

* Changed the document structure into a general transaction id part and one etag specific part.

* Renamed entag attribute to etag, prefix to txid, namespace to urn:ietf:params:xml:ns:yang:ietf-netconf-txid.

* Set capability string to urn:ietf:params:netconf:capability:txid:1.0

* Changed YANG module name, namespace and prefix to match names above.

* Harmonized/slightly adjusted etag value space with RFC 7232 and RFC 8040.

* Removed all text discussing etag values provided by the client (although this is still an interesting idea, if you ask the author)

* Clarified the etag attribute mechanism, especially when it comes to matching against non-versioned elements, its cascading upwards in the tree and secondary effects from when- and choice-statements.

* Added a mechanism for returning the server assigned etag value in get-config and get-data.

* Added section describing how the NETCONF discard-changes, copy-config, delete-config and commit operations work with respect to etags.

* Added IANA Considerations section.

* Removed all comments about open questions.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Acknowledgments

The author wishes to thank Benoit Claise for making this work happen, and the following individuals, who all provided helpful comments: Per Andersson, Kent Watsen, Andy Bierman, Robert Wilton, Qiufang Ma.

Author's Address

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 August 2022

Q. Wu
W. Song
Huawei
P. Liu
China Mobile
Q. Ma
Huawei
W. Wang
China Telecom

24 February 2022

Adaptive Subscription to YANG Notification
draft-wang-netconf-adaptive-subscription-09

Abstract

This document defines a YANG data model and associated mechanism enabling the subscriber's adaptive subscriptions to a publisher's event streams with various different period intervals to report updates. Applying these elements allows servers automatically adjust the rate and volume of telemetry traffic sent from a publisher to receivers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 August 2022.

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. Model Overview 4
2.1. Subscription Configuration 5
2.2. YANG RPC 7
2.2.1. "establish-subscription" RPC 7
2.3. Notifications for Adaptive Subscribed Content 8
3. XPath Complexity Evaluation 9
4. Adaptive Subscription YANG Module 10
5. IANA Considerations 14
5.1. Updates to the IETF XML Registry 14
5.2. Updates to the YANG Module Names Registry 14
6. Security Considerations 14
7. Contributors 15
8. Acknowledges 15
9. References 15
9.1. Normative References 15
9.2. Informative References 17
Appendix A. Example YANG Module 17
A.1. "example-wifi-mac" YANG Module 18
Appendix B. Adaptive Subscription and Notification Example . . . 22
B.1. "edit-config" Example 23
B.2. Create Adaptive Subscription Example 24
B.3. "xpath-evaluation-unsupported" error response example . . 24
B.4. "adaptive-period-update" notification example 25
B.5. Changes between Revisions 26
Authors' Addresses 28

1. Introduction

YANG-Push subscriptions [RFC8641] allow subscriber applications to request a continuous customized stream of updates from a YANG datastore without needing to poll. It defines a mechanism (i.e., update trigger) to determine when an update record needs to be generated. Two types of subscriptions are introduced in [RFC8641], distinguished by how updates are triggered: periodic and on-change.

* Periodic subscription allows subscribed data to be streamed to the destination at a configured fixed periodic interval;

* On-change subscription allows update to be triggered whenever a change in the subscribed information is detected.

However in some large scale deployments (e.g., wireless network performance monitoring) where an increased data collection rate is used, it becomes more likely that receivers are temporarily overwhelmed with a burst of streamed data and it also consumes expensive network resource (e.g., radio resource). If the rate at which we can collect a stream of data is set too low or chosen to get low priority telemetry data dropped, these telemetry data are not sufficient to detect and diagnose problems and verify correct network behavior.

There is a need for a service to configure both clients and servers with multiple different period intervals and corresponding subscription update policy which allows servers/publishers automatically switch to different period intervals according to server resource usage change without the interaction with the client for policy update instruction, e.g., when the wireless signal strength falls below a configured threshold, the subscribed data can be streamed at a higher rate while when the wireless signal strength crosses a configured threshold, the subscribed data can be streamed at a lower rate.

This document defines a YANG data model and associated mechanism enabling the subscriber's adaptive subscriptions to a publisher's event streams. Applying these elements allows servers automatically adjust the rate and volume of telemetry traffic sent from a publisher to receivers.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC5277] [RFC7950] [RFC3198] [RFC8342] [RFC8639] [RFC8641] and are not redefined here:

* Event

* Client

* Configuration
* Configured subscription
* Configuration datastore
* Notification message
* Publisher
* Receiver
* Subscriber
* Subscription
* On-change subscription
* Periodic subscription
* Selection filter

This document defines the following term:

Adaptive Subscription: Apply subscription update policy on the servers and allow servers/publishers automatically switch to different period intervals according to the server resource usage change without the interaction with the client for update policy instruction.

2. Model Overview

This document defines a YANG module "ietf-adaptive-subscription", which augments the "update-trigger" choice defined in the "ietf-yang-push" module [RFC8641] with subscription configuration parameters that are specific to a subscriber's adaptive subscription.

In addition to subscription state notifications defined in [RFC8639] and notifications for subscribed content defined in [RFC8641], "ietf-adaptive-subscription" YANG module also defines "adaptive-period-update" notification to report the update interval change.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-adaptive-subscription" module.

```
module: ietf-adaptive-subscription
  augment /sn:subscriptions/sn:subscription/yp:update-trigger:
    +--rw (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
              +--rw adaptive-period* [name]
                 +--rw name                      string
                 +--rw xpath-external-eval       string
                 +--rw period                    yp:centiseconds
                 +--rw anchor-time?              yang:date-and-time
  augment /sn:establish-subscription/sn:input/yp:update-trigger:
    +-- (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
             +--rw adaptive-period* [name]
                +--rw name                      string
                +--rw xpath-external-eval       string
                +--rw period                    yp:centiseconds
                +--rw anchor-time?              yang:date-and-time
  notifications:
    +---n adaptive-period-update
       +--ro id?                                      sn:subscription-id
       +--ro period                                   yp:centiseconds
       +--ro anchor-time?                             yang:date-and-time
       +--ro (selection-filter)?
          +--:(by-reference)
          |  +--ro selection-filter-ref               selection-filter-ref
          +--:(within-subscription)
             +--ro (filter-spec)?
                +--:(datastore-subtree-filter)
                |  +--ro datastore-subtree-filter?   <anydata> {sn:subtree}?
                +--:(datastore-xpath-filter)
                   +--ro datastore-xpath-filter?     yang:xpath1.0 {sn:xpath}?
```

2.1. Subscription Configuration

For adaptive subscriptions, triggered updates will occur at the boundaries of specified time intervals when a trigger condition is satisfied. These boundaries can be calculated from the following adaptive periodic parameters:

* a "name" represents the name of each adaptive period;

* a "period" that defines the new duration between push updates, the period can be switched based on trigger conditions;

* an "anchor-time"; update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

* an "xpath-external-eval" represents a standard XPath Evaluation criterion (See section 6.4 of [RFC7950]) that is applied against the targeted data object, which is used to trigger/control the update interval switching within the server. It follows the rules defined in section 3.4 of [XPATH1.0] and contains comparisons of the targeted datastore node with its value to the specific threshold in the XPath format. Different from selection filter defined in [RFC8641],

  - it is applied against a single targeted object rather than a set of target objects.

  - it monitors a specific data object change and evaluates the trigger condition associated with the targeted object to be true or false using XPATH rules and doesn't influence the event records output generation from a publisher.

  The targeted object can be evaluated in the returned node set at the end of streaming update period. To reduce the frequency of evaluation, the server can choose to check targeted object change at every multiple (e.g.,2 or 3) update periods.

  The represented expression defined in "xpath-external-eval" is evaluated in the following XPath context:

  - The set of namespace declarations is the set of prefix and namespace pairs for all YANG modules implemented by the server, where the prefix is the YANG module name and the namespace is as defined by the "namespace" statement in the YANG module.

  - If the leaf is encoded in XML, all namespace declarations in scope on the "xpath-external-eval" leaf element are added to the set of namespace declarations. If a prefix found in the XML is already present in the set of namespace declarations, the namespace in the XML is used.

  - The set of variable bindings is empty.

  - The function library is the core function library defined in [XPATH1.0] and the function defined in Section 10 in RFC 7950.

  - The context node is the root node.

For the cases where multiple list instances are needed to handle in "xpath-external-eval", XPath abbreviated syntax can be used to identify a particular instance, e.g., to represent a comparison for a leaf in a list entry:

/if:interfaces/if:interface[if:name="eth0"]/if:in-errors>1000.

The server MUST convert the XPath expression defined in "xpath-external-eval" to a boolean value and internally apply the "boolean" function defined in Section 4.3 in [XPATH1.0] if the evaluated result is not a boolean.

Note that the adaptive subscription may not be supported by every YANG datastore nodes. A publisher MAY decide to simply reject an adaptive subscription with "adaptive-unsupported" if the scope of the subscription contains selected data nodes for which adaptive subscription is not supported.

2.2. YANG RPC

2.2.1. "establish-subscription" RPC

The augmentation of YANG module "ietf-yang-push" made to RPCs specified in YANG module "ietf-subscribed-notifications" [RFC8639] is introduced. This augmentation concerns the "establish-subscription" RPC, which is augmented with parameters that are needed to specify a subscriber's adaptive subscriptions. These parameters are same as one defined in Section 2.1.

2.2.1.1. RPC Failures

As specified in [RFC8639] and [RFC8641], RPC error responses from the publisher are used to indicate a rejection of an RPC for any reason. This document introduces three new RPC errors for "establish-subscription" RPC.

establish-subscription
---
adaptive-unsupported
xpath-evaluation-unsupported
multi-xpath-criteria-conflict

Adaptive-unsupported is used to indicate that the adaptive subscription is not supported for any objects that are selectable by the filter.

Xpath-evaluation-unsupported is used to indicate that a server fails to parse syntax defined in "xpath-external-eval". The failure can be caused by either a syntax error or some XPath 1.0 syntax not supported against the specific object.

Multi-xpath-criteria-conflict is used to indicate that the multiple Xpath evaluation criteria represented by "xpath-external-eval" is evaluated as conflict, i.e., more than one condition expressions are evaluated to "true". Such a conflict may also cause an ongoing adaptive-subscription terminated.

For an example of how above RPC errors can be returned, see the "xpath-evaluation-unsupported" error response illustrated in Appendix B.3.

2.3. Notifications for Adaptive Subscribed Content

The adaptive update notification is similar to subscription state change notifications defined in [RFC8639]. It is inserted into the sequence of notification messages sent to a particular receiver. The adaptive update notification cannot be dropped or filtered out, it cannot be stored in replay buffers, and it is delivered only to impacted receivers of a subscription. The identification of adaptive update notification is easy to separate from other notification messages through the use of the YANG extension "subscription-state-notif". This extension tags a notification as a subscription state change notification.

The objects in the 'adaptive-period-update' notification include:

* a "period" that defines the duration between push updates, the period can be changed based on trigger conditions.

* an "anchor-time"; update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

* A selection filter identifying YANG nodes of interest in a datastore. Filter contents are specified via a reference to an existing filter or via an in-line definition for only that subscription based on XPath Evaluation criteria defined in section 6.4 of [RFC7950]. Referenced filters allow an implementation avoid evaluating filter acceptability during a dynamic subscription request. The "case" statement differentiates the options. Note that filter contents are not affected by "xpath-external-eval" parameter defined by the update trigger.

3. XPath Complexity Evaluation

YANG-Push subscriptions [RFC8641] specifies selection filters to identify targeted YANG datastore nodes and/or datastore subtrees for which updates are to be pushed. In addition, it specifies update policies which contain conditions that trigger generation and pushing of new update records. To support a subscriber's adaptive subscription defined in this document, the trigger condition can also use similar selection filters to express a standard XPath Evaluation criterion (section 6.4 of [RFC7950]) against targeted data objects.

Similar to on-change subscriptions, adaptive subscriptions are particularly effective for data that changes infrequently, the following complex design choices need to be cautious, although these designs have already been well supported by the section 3.4 of [XPATH1.0]:

* Support XPath Evaluation criteria against every data object;

* Support any type of node set in the XPath Evaluation criterion, e.g., string,int64, uint64, and decimal64 types;

* Both objects in the XPath Evaluation criterion to be compared are node-sets;

* Two objects to be compared are in different data types, e.g., one is integer, the other is string

As described in section 6.4 of RFC7950, Numbers in XPath 1.0 are IEEE 754 [IEEE754-2008] double-precision floating-point values; some values of int64, uint64, and decimal64 types cannot be exactly represented in XPath expressions.

If two objects to be compared are in different data types, conversion function is needed to convert different data types into numbers.

If both objects in XPath Evaluation criteria to be compared are node-sets, more computation resources are required which add complexity.

To reduce these complexities, the following design principles are recommended:

* XPath Evaluation criteria against a minimal set of data objects in the data model, the minimal set of data objects can be advertised using Notification capabilities model defined in [RFC9196].

* XPath Evaluation criteria only support condition expressions that filter updates based on numbers.

* One object to be compared in the XPath Evaluation criteria MUST be a leaf data node.

* The other object to be compared in the XPath Evaluation criteria MUST be number data type.

If a server receives an XPath Evaluation criterion with some XPath syntax unsupported against the specific object, an RPC error with "xpath-evaluation-unsupported" should be returned.

4. Adaptive Subscription YANG Module

<CODE BEGINS> file "ietf-adaptive-subscription@2020-02-14.yang"

```
module ietf-adaptive-subscription {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription";
  prefix as;
  import ietf-subscribed-notifications {
    prefix sn;
  }
  import ietf-yang-push {
    prefix yp;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "";
  description
    "NETCONF Protocol Data Types and Protocol Operations.
     Copyright (c) 2020 IETF Trust and the persons identified as
     the document authors.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC xxxx; see
     the RFC itself for full legal notices.";

  revision 2020-02-14 {
    description
```

```
        "Initial revision";
      reference
        "RFCxxx Adaptive subscription to YANG notification.";
    }

    identity adaptive-unsupported {
      base sn:establish-subscription-error;
      description
        "Adaptive-subscription is not supported for any objects
         that are selectable by the filter.";
    }

    identity xpath-evaluation-unsupported {
      base sn:establish-subscription-error;
      description
        "Unable to parse the xpath evaluation criteria defined in
        xpath-external-eval because of a syntax error or some
         XPath 1.0 syntax not supported against the specific object.";
    }

    identity multi-xpath-criteria-conflict {
      base sn:establish-subscription-error;
      base sn:subscription-terminated-reason;
      description
        "Multiple Xpath evaluation criteria represented by
         'xpath-external-eval' is evaluated as conflict, i.e.,
         more than one condition expressions are evaluated to
        'true'.";
  }

    grouping adaptive-subscription-modifiable {
      description
        "This grouping describes the datastore-specific adaptive subscription
         conditions that can be changed during the lifetime of the
         subscription.";
      choice adaptive-subscription {
        description
          "Defines necessary conditions for sending an event record to
                 the subscriber.";
        container adaptive-subscriptions {
          list adaptive-period {
            key "name";
            description
              "Defines necessary conditions to switch update interval for
              sending an event record to the subscriber. The event record output
              generation will not be influenced these conditions.";
            leaf name {
              type string {
```

```
            length "1..64";
             }
          description
            "The name of the condition to be matched.  A device MAY further
             restrict the length of this name; space and special
             characters are not allowed.";
           }
            leaf xpath-external-eval {
              type string;
              description
                "A XPath string, representing a logical expression,
                 which can contain comparisons of datastore values
                 and logical operations in the XPath format.";
            }
            leaf period {
              type yp:centiseconds;
              mandatory true;
              description
                "Duration of time that should occur between periodic
                 push updates, in units of 0.01 seconds.";
            }
            leaf anchor-time {
              type yang:date-and-time;
              description
                "Designates a timestamp before or after which a series
                 of periodic push updates are determined.  The next
                 update will take place at a point in time that is a
                 multiple of a period from the 'anchor-time'.
                 For example, for an 'anchor-time' that is set for the
                 top of a particular minute and a period interval of a
                 minute, updates will be sent at the top of every
                 minute that this subscription is active.";
            }
          }
          description
            "Container for adaptive subscription.";
        }
      }
    }

    augment "/sn:subscriptions/sn:subscription/yp:update-trigger" {
      description
        "This augmentation adds additional subscription parameters
         that apply specifically to adaptive subscription.";
      uses adaptive-subscription-modifiable;
    }
    augment "/sn:establish-subscription/sn:input/yp:update-trigger" {
      description
```

```
        "This augmentation adds additional subscription parameters
           that apply specifically to datastore updates to RPC input.";
      uses adaptive-subscription-modifiable;
    }

    notification adaptive-period-update {
      sn:subscription-state-notification;
      description
        "This notification contains a push update that in turn contains
         data subscribed to via a subscription.  In the case of a
         periodic subscription, this notification is sent for periodic
         updates.  It can also be used for synchronization updates of
         an on-change subscription.  This notification shall only be
         sent to receivers of a subscription.  It does not constitute
         a general-purpose notification that would be subscribable as
         part of the NETCONF event stream by any receiver.";
      leaf id {
        type sn:subscription-id;
        description
          "This references the subscription that drove the
           notification to be sent.";
      }
      leaf period {
        type yp:centiseconds;
        mandatory true;
        description
          "New duration of time that should occur between periodic
           push updates, in units of 0.01 seconds.";
      }
      leaf anchor-time {
        type yang:date-and-time;
        description
          "Designates a timestamp before or after which a series
           of periodic push updates are determined.  The next
           update will take place at a point in time that is a
           multiple of a period from the 'anchor-time'.
           For example, for an 'anchor-time' that is set for the
           top of a particular minute and a period interval of a
           minute, updates will be sent at the top of every
           minute that this subscription is active.";
      }
      uses yp:datastore-criteria {
        refine "selection-filter/within-subscription" {
          description
            "Specifies the selection filter and where it originated
             from.  If the 'selection-filter-ref' is populated, the
             filter in the subscription came from the 'filters'
             container.  Otherwise, it is populated in-line as part
```

```
              of the subscription itself.";
          }
        }
      }
    }
    <CODE ENDS>
```

5. IANA Considerations

5.1. Updates to the IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

```
--------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.
--------------------------------------------------------------------
```

5.2. Updates to the YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC7950]. Following the format in [RFC6020], the following registration is requested to be made:

```
--------------------------------------------------------------------
   Name:         ietf-adaptive-subscription
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
   Prefix:       as
   Reference:    RFC xxxx
--------------------------------------------------------------------
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

* /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:period

* /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:anchor-time

* /sn:establish-subscription/sn:input/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:period

* /sn:establish-subscription/sn:input/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:anchor-time

7. Contributors

Thanks Michale Wang, Liang Geng for their major contributions to the initial modeling and use cases.

Michale Wang
Email: wangzitao@huawei.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

8. Acknowledges

We would like to thanks Rob Wilton, Thomas Graf, Andy Bierman, Michael Richardson, Henk Birkholz for valuable review on this document, special thanks to Thomas and Michael to organize the discussion on several relevant drafts and reach the common understanding on the concept and ideas. Thanks Michael for providing CHIP/Matter WIFI statistics reference.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

[RFC9196] Lengyel, B., Clemm, A., and B. Claise, "YANG Modules Describing Capabilities for Systems and Datastore Update Notifications", RFC 9196, DOI 10.17487/RFC9196, February 2022, <https://www.rfc-editor.org/info/rfc9196>.

9.2. Informative References

[CHIP] CSA, "Connected Home over IP Specification", April 2021.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[XPATH1.0] W3C, "https://www.w3.org/TR/1999/REC-xpath-19991116/", 11 November 1999.

Appendix A. Example YANG Module

The example YANG module used in this document represents a Wi-Fi Network Diagnostics data specified in [CHIP] which can be used by a Node to assist a user or Administrative Node in diagnosing potential problems.

YANG tree diagram for the "example-wifi-network-diagnostic" module:

```
module: example-wifi-network-diagnostic
  +--rw server
  |  +--rw bssid?                        yang:mac-address
  |  +--rw security-type?                enumeration
  |  +--rw wifi-version?                 enumeration
  |  +--rw channel-num?                  int8
  |  +--rw rssi?                         int8
  |  +--rw beacon-lost-count?            int8
  |  +--rw beacon-rx-count?              int8
  |  +--rw packet-multicast-rx-count?    int8
  |  +--rw packet-multicast-tx-count?    int8
  |  +--rw packet-unicast-rx-count?      int8
  |  +--rw packet-unicast-tx-count?      int8
  |  +--rw current-max-rate?             int8
  |  +--rw overrun-count?                int8
  +--rw events
     +--rw event* [name]
        +--rw name                    string
        +--rw disconnection?          enumeration
        +--rw association-failure?    enumeration
        +--rw connection-status?      enumeration
```

A.1. "example-wifi-mac" YANG Module

```
module example-wifi-network-diagnostic {
  yang-version 1;
  namespace "http://example.com/yang/wifi-network-diagnostic";
  prefix wnd;

  import ietf-yang-types {
    prefix yang;
  }

  container server {
    description
      "Configuration of the WiFi Server logical entity.";
    leaf bssid {
      type yang:mac-address;
      description
        "The MAC address of a wireless access point.";
    }
    leaf security-type {
      type enumeration {
        enum unspecified {
          value 0;
        }
        enum none {
          value 1;
```

```
          }
          enum wep {
            value 2;
          }
          enum wpa {
            value 3;
          }
          enum wpa2 {
            value 4;
          }
          enum wpa3 {
            value 5;
          }
        }
        description
          "The type of Wi-Fi security used. A value of 0
           indicate that the interface is not currently
           configured or operational.";
      }
      leaf wifi-version {
        type enumeration {
          enum 80211a {
            value 0;
          }
          enum 80211b {
            value 1;
          }
          enum 80211g {
            value 2;
          }
          enum 80211n {
            value 3;
          }
          enum 80211ac {
            value 4;
          }
          enum 80211ax {
            value 5;
          }
        }
        description
          "The highest 802.11 standard version usable
           by the Node.";
      }
      leaf channel-num {
        type int8;
        description
          "The channel that Wi-Fi communication is currently
```

```
          operating on. A value of 0indicates that the interface
          is not currently configured or operational.";
      }
      leaf rssi {
        type int8;
        description
          "The RSSI of the Nodes Wi-Fi radio in dBm.";
      }
      leaf beacon-lost-count {
        type int8;
        description
          "The count of the number of missed beacons the
           Node has detected.";
      }
      leaf beacon-rx-count {
        type int8;
        description
          "The count of the number of received beacons. The
           total number of expected beacons that could have been
           received during the interval since association SHOULD
           match the sum of BeaconRxCount and BeaconLostCount. ";
      }
      leaf packet-multicast-rx-count {
        type int8;
        description
          "The number of multicast packets received by
           the Node.";
      }
      leaf packet-multicast-tx-count {
        type int8;
        description
          "The number of multicast packets transmitted by
           the Node.";
      }
      leaf packet-unicast-rx-count {
        type int8;
        description
          "The number of multicast packets received by
           the Node.";
      }
      leaf packet-unicast-tx-count {
        type int8;
        description
          "The number of multicast packets transmitted by
           the Node.";
      }
      leaf current-max-rate {
        type int8;
```

```
        description
          "The current maximum PHY rate of transfer of
           data in bytes-per-second.";
      }
      leaf overrun-count {
        type int8;
        description
          "The number of packets dropped either at ingress or
           egress, due to lack of buffer memory to retain all
           packets on the ethernet network interface. The
           OverrunCount attribute SHALL be reset to 0 upon a
           reboot of the Node..";
      }
    }
    container events {
      description
        "Configuration of WIFI Network Diagnostic events.";
      list event {
        key "name";
        description
          "The list of event sources configured on the
           server.";
        leaf name {
          type string;
          description
            "The unique name of an event source.";
        }
        leaf disconnection {
          type enumeration {
            enum de-authenticated {
              value 1;
            }
            enum dis-association {
              value 2;
            }
          }
          description
            "A Nodes Wi-Fi connection has been disconnected as a
             result of de-authenticated or dis-association and
             indicates the reason.";
        }
        leaf association-failure {
          type enumeration {
            enum unknown {
              value 0;
            }
            enum association-failed {
              value 1;
```

```
            }
            enum authentication-failed {
              value 2;
            }
            enum ssid-not-found {
              value 3;
            }
          }
          description
            "A Node has attempted to connect, or reconnect, to
             a Wi-Fi access point, but is unable to successfully
             associate or authenticate, after exhausting all
             internal retries of its supplicant.";
        }
        leaf connection-status {
          type enumeration {
            enum connected {
              value 1;
            }
            enum notconnected {
              value 2;
            }
          }
          description
            "A Node's connection status to a Wi-Fi network has
             changed. Connected, in this context, SHALL mean that
             a Node acting as a Wi-Fi station is successfully
             associated to a Wi-Fi Access Point.";
        }
      }
    }
  }
```

Appendix B. Adaptive Subscription and Notification Example

The examples within this document use the normative YANG module "ietf-adaptive-subscription" defined in Section 4 and the non-normative example YANG module "example-wifi-network-diagnostic" defined in Appendix A.1.

This section shows some typical adaptive subscription and notification message exchanges.

B.1. "edit-config" Example

The client configures adaptive subscription policy parameters on the server. The adaptive subscription configuration parameters require the server to support two update intervals (i.e., 5 seconds, 60 seconds) and report updates every 60 seconds if the rssi value is greater than or equal to -65dB; If the rssi value is less than -65dB, switch to 5 seconds period value to report updates.

```
<rpc message-id="101"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
  <config
   xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <top
    xmlns="http://example.com/schema/1.2/config"
    xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <yp:datastore
     xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
             ds:running
    </yp:datastore>
    <yp:datastore-xpath-filter
     xmlns:wnd="https://example.com/sample-data/1.0">
            /wnd:example-wifi-network-diagnostic
    </yp:datastore-xpath-filter>
    <as:adaptive-subscriptions
     xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
     <as:adaptive-period>
      <as:xpath-external-eval>
       /wnd:server/wnd:rssi&lt;-65
      </as:xpath-external-eval>
      <as:period>5</as:period>
     </as:adaptive-period>
     <as:adaptive-period>
      <as:xpath-external-eval>
       /wnd:server/wnd:rssi&gt;=-65
      </as:xpath-external-eval>
      <as:period>60</as:period>
    </as:adaptive-period>
    </as:adaptive-subscriptions>
   </top>
  </config>
 </edit-config>
</rpc>
```

B.2. Create Adaptive Subscription Example

The subscriber sends an "establish-subscription" RPC with the parameters listed in to request the creation of an adaptive subscription. The adaptive subscription configuration parameters require the server to report updates every 5 seconds if the rssi value is less than -65dB; If the rssi value is greater than or equal to -65dB, switch to 60 seconds period value. (Section 2)

```
<netconf:rpc message-id="101"
 xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
 <establish-subscription
  xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
  xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
  <yp:datastore
   xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
         ds:running
  </yp:datastore>
  <yp:datastore-xpath-filter
   xmlns:wnd="https://example.com/sample-data/1.0">
       /wnd:example-wifi-network-diagnostic
  </yp:datastore-xpath-filter>
  <as:adaptive-subscriptions
   xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
   <as:adaptive-period>
    <as:xpath-external-eval>
     wnd:server/wnd:rssi&lt;-65
    </as:xpath-external-eval>
    <as:period>5</as:period>
   </as:adaptive-period>
   <as:adaptive-period>
    <as:xpath-external-eval>
     wnd:server/wnd:rssi&gt;=-65
    </as:xpath-external-eval>
    <as:period>60</as:period>
   </as:adaptive-period>
  </as:adaptive-subscriptions>
 </establish-subscription>
</netconf:rpc>
```

B.3. "xpath-evaluation-unsupported" error response example

If the subscriber has authorization to establish the subscription with a server, but the server had not been able to fully satisfy the request from the subscriber, the server should send an RPC error response.

For instance, if the XPATH 1.0 syntax against the targeted data object defined in "xpath-external-eval" is not supported by the server' s implementation, the server returns a reply indicating a failure. The following <rpc-reply> illustrates an example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-app-tag>
      ietf-adaptive-subscription:xpath-evaluation-unsupported
    </error-app-tag>
    <error-path xmlns:wnd="https://example.com/sample-data/1.0">
      /wnd:server/wnd:rssi
    </error-path>
  </rpc-error>
</rpc-reply>
```

Since adaptive subscription allows a server to be configured with multiple different period intervals and corresponding XPath evaluation criteria to trigger update interval switch in the server, it may be possible for the server to return multiple <rpc-error> elements with "xpath-evaluation-unsupported" failure specified by different error paths. The subscriber can use this information in future attempts to establish a subscription.

B.4. "adaptive-period-update" notification example

Upon the server switches from the update interval 5 seconds to the new update interval 60 seconds, before sending event records to receivers, the "adaptive-period-update" notification should be generated and sent to the receivers to inform the receivers that the update interval value is switched to the new value.

```
<notification
 xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
 xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
 <eventTime>2016-11-21T13:51:00Z</eventTime>
 <adaptive-period-update
  xmlns="http://example.com/ietf-adaptive-subscription">
  <id>0</id>
  <period>60</period>
  <yp:datastore
   xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:running
  </yp:datastore>
  <yp:datastore-xpath-filter
   xmlns:ex="https://example.com/sample-data/1.0">
       /ex:example-wifi-network-diagnostic
  </yp:datastore-xpath-filter>
 </adaptive-period-update>
</notification>
```

B.5. Changes between Revisions

v08 -v09

* Define two new RPC errors to report when adaptive subscription unsupported or multiple XPath criteria conflict.

* Remove the "watermark" parameter.

* Add clarification about how to evaluate the XPath expression defined in "xpath-external-eval".

* Add clarification about how to compare a targeted data object in a specific list entry.

v07 -v08

* Define a new RPC error to report when an XPath syntax defined in "xpath-external-eval" is unsupported by a server.

* Add a new example showing how the RPC error being returned by a publisher.

* The usage examples fixed in the Appendix.

* Grammatical errors correction(missing articles, plurality mismatches, etc).

v06 -v07

* The usage examples typo fixed in the Appendix.

* Add reference to RFC7950 XPATH Evaluation section and XPATH 1.0

* Clarify the definitions of 'xpath-external-eval' and 'selection-filter' by reusing XPATH Evaluation rules in RFC7950.

* Add a new terminology "adaptive subscription".

* Add one section to discuss Arbitrary XPath Complexity.

v05 -v06

* Replace example-wifi-mac module with example-wifi-network-diagnostic using WIFI statistics specified in CHIP specification.

* Update adaptive subscription Example to align with WIFI example module change.

* Add one more reference to CHIP Specification.

v04 -v05

* Remove "modify-subscption" RPC usage.

* Module update to fix the nits.

* Update adaptive subscription Example.

* Other Editorial changes.

v03 - v04

* Add missing subtrees and data nodes in the security section;

* Change "adaptive-update" notification into "adaptive-period-update" notification;

* Other Editorial changes.

v02 - v03

* Clarify the difference between low priority telemetry data dropping and collection rate switching in the introduction section;

* Update the abstract and introduction section to focus on collection rate switching in the server without interaction with the remote client;

* Format usage example and change ssid into rssi in the appendix;

* Use boilerplate and reuse the terms in the terminology section.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Wei Song
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: songwei80@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing
Email: liupengyjy@chinamobile.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

Wei Wang
China Telecom
32 Xuanwumen West St, Xicheng District
Beijing

Email: wangw36@chinatelecom.cn